DRAGON USER

January 1987

The independent Dragon magazine

Contents

Letters 2

More machine code books from Pam D'Arcy ... a Cardiff users group...how to double your single sided discs...DOS warning...Chuckie Egg hiscore.

People's Chart 3

A roundup of the best selling programs from the past ten months of voting.

News desk 4

OS-9 licence given to H.C. Andersen software... The Thirteenth Task from Arc Software...new progs from Microvision...Quickbeam take over Smithson Computing... Christmas offers from Buzz Software.

The Answer 5

By popular request, Dragon User will from now on be publishing Gordon Lee's own solution to the competition which we are judging in the current month.

Daisy Wheel 6

Mike Hides' method for using an ordinary daisywheel printer as a graphics device.

Fractal pictures 8

All the trees are bare, and the snowflakes are falling. At least, they are inside Brian Hulley's Dragon, where self-replicating graphic designs are expanding to fill the screen.

Dragon Soft 12

6809 Express and Shaolin Master from Quickbeam ... Logo 3.4 from Bernd Knechtel Software.

Inside The 32 13

A blockbuster look at the memory and processing structure of the Dragon 32, by Dave Barnish B.Sc.

Shorts 20

Two short programs: Intelligent Pattern Generator and Memory Browser

Expert's Arcade Arena 21

The Expert takes a few pokes at his hapless followers, and drops even more hints about his Christmas present.

Dragon Answers 23

Brian Cadge takes a look at ATN functions...RESTORE in Dragon Basic...printer lead pin outs...intercepting printer output...changing disc drives without stopping the program.

Winners and Losers 24

Gordon Lee looks at some fine examples of the right and wrong way to win a DU puzzle.

Adventure Trail 26

Mike Gerrard hits the trail again, this time with the two gentlemen of Verona, and Fernando Poo straight from the author.

Competition 30

Gordon Lee gets designs on the Dragon with a challenge to come up with the most interesting repeating pattern.

Editorial

A last, quick Happy Christmas to you all if this reaches you when it should, and if it doesn't, then a Happy New Year! I had such a 'good' Christmas last year that I'm thinking of keeping it all to a minimum this year — so you may think of me crouched over my word processor in those snowy, silent days between 25th December and January 1st, patting the next issue into shape. On the other hand ... don't phone. I might be at a party!

There are one or two pieces of interesting news about ... H. C. Andersen of Denmark have been given a licence to sell OS-9 in Europe — see News Desk for details. And Quickbeam Software have taken over Smithson Computing's software list. The recently silent Microvision has re-emerged with a whole new list of utilities and a few games as well.

In response to many suggestions, Gordon Lee has been roped in to double his input to Dragon User by extracting some of the better and worse solutions to his puzzles and analysing their success or failure. This month Gordon outlines the principles of effective puzzle-solving. We will also be printing his solution to the current month's prizewinning efforts in future.

Because of a last minute let down, none of the Dragon reporters made it to the London 6809 Show — we'd like personal reports from anyone who had a good look round. The best ones'll get paid and printed.

Telephone number
(All departments)
437-4343

Editor
HELEN ARMSTRONG

Production Editor
BARBORA HÁJEK

Associate Editor
JOHN COOK

Editorial Secretary
ANNE MARIE O'DWYER

Advertisement Manager
RODNEY WOODLEY

Administration
GERALDINE SMYTH

Marketing Manager
HELEN PERRY

Managing Editor
PETER WORLOCK

Publishing Director
JENNY IRELAND

Subscriptions
UK £14 for 12 issues
Overseas (surface) £20 for 12 issues
ISSN 0265-177. Telex: 296275
Dragon User, 12/13 Little Newport Street, London WC2H 7PP
US address: c/o Business Press International, 205 East 42nd St, New York, NY 10017
Published by Scot Press Ltd.

Typesetting and Production by Artext Limited, London NW1.
Printed by Headley Brothers Ltd. Ashford, Kent
Registered at the Post Office as a newspaper.

How to submit articles

The quality of the material we can publish in *Dragon User* each month will, to a very great extent depend on the quality of the discoveries that you can make with your Dragon. The Dragon computer was launched on to the market with a powerful version of Basic, but with very poor documentation.

Articles which are submitted to *Dragon User* for publication should not be more than 3000 words long. All submissions should be typed. Please leave wide margins and a double space between each line. Programs should, whenever possible, be computer printed on plain white paper and be accompanied by a tape of the program.

We cannot guarantee to return every submitted article or program, so please keep a copy. If you want to have your program returned you must include a stamped addressed envelope.

Letters

This is the chance to air your views — send your tips, compliments and complaints to Letters Page, *Dragon User*, 12-13 Little Newport Street, London WC2H 7PP.

Guilt edged

IN OCTOBER 1986 issue you reviewed a program from Cascade Games called Cassette 50.

I purchased this cassette seven months ago and must entirely agree with your review. In fact I'm surprised you gave it even one dead dragon.

I have to say that I feel guilty about not writing to you at the time to warn people not to buy this complete waste of money, as they say, there's one born every minute!

However, not all is bad, as you reviewed Electronic Author. I myself have Printer Control from MacGowan Consultants which is in my opinion an excellent piece of software which I would have no hesitation in recommending. Your review in a previous issue did not really touch on the excellence and real value for money of this program. Top marks from me.

By the way, apparently CumanaDOS 2.0 has several severe bugs in it, or so I am informed. If you know about these then the message should have been passed on.

Keith Gardener
81 Tewkesbury Street
Tudor Road
Leicester LE3 5HQ

THE EDITOR has had no first hand complaints about CumanaDOS, but one or two enquiries, and would like to hear from someone who has used the latest version thoroughly ...

Teacher's pet

I HAVE been buying Dragon User since the first issue in 1983, and now have a subscription.

But what do I find? Articles which treat users as morons. Surely there cannot be many (if any) users of the Dragon left who do not know how to program in Basic! The standard of the programs published has not changed in any way since the first issue. I do hope this will be improved, as the machine code articles and programs are of a very good standard. As a computer studies lecturer I see enough poorly thought out pieces of work.

A. D. Butler
Flat 5
18 Merridale Lane
Merridale
Wolverhampton

NOT AT all. Not only are new users still coming to the Dragon, as people pick them up at bargain prices, but we have a steady demand for Basic programs, as well as machine code.

As a teacher with experience of correcting awkward pieces of programming, how about writing an educational piece based on your experiences? Drop us a line.

Cardiff club

CALLING all Dragon and Tandy computer users in the Cardiff area, interested in joining a User Group? If so, telephone Mr. J. Jones on Cardiff (0222) 553559.

J. Jones
43 Firs Ave.
Pentrbane
Cardiff CF5 3TF

Double your disc

HAVING glanced through the July issue I read with interest the article on disc drives by Geoff Ettridge. He mentioned that with the Dragon Drive using double sided discs, only one side can be used. THis is true using the disc as purchased, but both sides can be used if one uses a paper punch and punches another small hole in the cover on the other side of the centre line to the existing one, keeping the same geometrical distance from the centre. The holes on both sides must be opposite each other for the beam to pass through. Care must be taken so as not to damage the disc. A notch must also be cut out on the opposite edge, and in a similar position, to the existing square notch.

I have over a dozen such discs operating in this way with no problems whatsoever. I trust this information can be passed on to the publishing editor for the benefit of other Dragon Drive users.

L. F. Forte
P.O. Box 196
Umkomaas 4170
Rep. of South Africa

A week point

I MUST commend you on getting something right. No, I'm not referring to your subscription-only policy. I'm referring to your getting my copy of DU here by the second day of the cover month! The June issue actually got here about a week after the May issue. I assume you've taken over from your former distributors. I felt that the problem was there all along and hope this arrangement will continue. I like your magazine very much (almost as much as my Dragon), but was not very happy with the sometime distribution. Your advertisers and writers with whom I've communicated have been exceptionally helpful, too. Thank you.

Donald Hicks
355 St. Emanuel St.
Mobile
Alabama AL 36603
USA

THE TRADITION is being preserved, rest assured, Donald. Quite a few of our readers got their September issue only a week after their October one ... the mailing company tell us that they think the labelling database is now sorted correctly, and Anne Marie, who runs the mailing list here, has been awarded an Executive Carton of Hedex.

Passed his eggsams

I AM desperate to know what the high score for Chuckie Egg is. The highest score I have ever got is 158870 level 17 after my dad mended the computer, he replaced the SAM chip. Level 17 has got five ducks and one mad duck.

Jonathan Baker
Aged 9

Late bookings

MANY thanks to the 'book spotters' who responded to my article (September issue). Additional titles to add to the convenient space at the bottom of page 14 *(Careful! - Ed.)* are:

The 6809 Companion (Babani) James, Microguide for the Dragon (Virgin/Interface) Gifford, 35 Programs for the Dragon32 (Century) Langdell, Color Computer Assembly Programming (Radio Shack) Barden, Getting Started on the TRS80/Dragon (Futura) Hartnell & Young.

And a correction: Hot Programs to Feed your Dragon (Sigma)is by Robinson, Smith and Blacow, not James.

The comment about Colour Computer Assembly Programming (Tandy catalogue number 62-2077) is that it uses, as you would expect, the EDTASM+ editor/assembler but 'is informative and readable'. Barden's 'Colour Computer Graphics' is the one I leap to for instant graphics info, so I can believe that.

Two comments were received on David Barrow's book: 1) 'generally find it good; an easier intro than Leventhal, though less detail' 2) 'I did not get a single program to work in just about the whole look and I found it too difficult'.

...you pays your money...

Not possessing the book, I couldn't help the reader who asked about the Draughts program in Enter The Dragon. Apparently it crashes when it makes the first move, but the fault cannot be found. Can anyone help him through this page, please?

Many thanks to those who took the trouble to write, either to me or to Dragon User, with further book news.

Pam D'Arcy
21 Wooburn Lane
Wooburn Green
High Wycombe
Bucks HP10 0HD

Dragon User People's Chart

The Best Games of 1986

THE final People's Chart ends with a roundup of the games you have voted the most popular in the last ten months. And we have some new plans.

1 Juxtaposition................(Wintersoft)

2 Shocktrooper................(Microdeal)

3 Shaolin Master............(Quickbeam)

4 Bean Stalker..............(Micro Vision)

5 Speed Racer..................(Microdeal)

THIS IS the Big One — the top five games of 1986, according to your votes. We have a surprise number one — Juxtaposition! Well, you'd be surprised if you never saw Dragon User in your life before — not other wise. Our mailbag, as well as the People's Chart, has borne bulk witness to the Big J's popularity. It's a shame and a pity that Wintersoft had to pack up before they could come up with part II.

The other top four favourites have all pretty well ruled the roost since the People's Chart began in March 1986, hotly pursued by other classics like Jet Set Willy, Eddie Steady Go, Rommel 3D, Moon Cresta, Total Eclipse (etc.)

This is also the Final Countdown. As features like the Expert's Arcade Arena and Gordon Lee's puzzle page grow in popularity, we have decided to hand over the opinions of the People to the people with their ears to the ground — The Expert and Mike Gerrard, on arcade games and adventures respectively. And we'll be making more room for comments from games players on the Letters Page, with star prizes for the best letter every month.

So thank you from us and from the software authors to everyone who voted in the People's Chart, keep playing and let us know what you find in the Dragon games world.

The final anagram has to be a good one — well, we think, its a good one. It goes "Take it from me — Dragon User is Best", and its from S. Goode of Sudbury! Your Microdeal software is on its way.

News desk

If you have any new products for the Dragon — software or hardware — ring the News Desk on 01-437 4343

OS-9 licence

DANISH suppliers H. C. Andersen Computer A/S have been licensed by Microware in the USA to distribute the OS-9 operating system, Level 1, Version 1.2 for the Dragon.

According to H. C. Anderson, they are the only company to be licensed by Microware to distribute OS-9 for the Dragon.

The Dragon OS-9 system includes edit and assembler, with English manuals, and will cost £174.00, inc. VAT but excluding delivery charges.

According to Microware, Thomson SIMIV of Paris has licensed the OS-9/68000 operating system for the software nucleus of the recently announced European Education Standard microcomputer system.

Three major European electronics companies, Thomson in France, Olivetti in Italy and Acorn in the UK are co-operating in developing a new standard in personal and educational micros.

The selection of OS-9 signifies its increasing acceptance as as a standard operat ing system for 68000 based micros. In late July, Tandy released the Color Computer III based on Microware's system.

Anyone wishing for more information should contact H.C. Andersen Computer A/S at Englandsvej 380, DK-2770 Kastrup, Denmark.

New, new, new

MICROVISION SOFTWARE, best known to Dragon User readers for reviewer Roy Coates, and Bean Stalker, have announced a new product list of unexpected extensiveness, including games, utilities and support programs, and Flex software.

'All the FLEX software comes on disc with documentation, and all the games are available on cassette or DragonDOS disc with the exception of *Catacomb Crisis*, which is currently only available on cassette' says Roy Coates.

The forthcoming list includes the aforementioned *Catacomb Crisis* a graphic adventure in caverns and tunnels, avoiding sparks and other nasties while looking for swords, cream cakes (these people have the right idea) and other useful items (£7.95); *Starship Destiny, Dungeon Destiny and Wild West Destiny*, three more graphic adventures with location graphics, problems and puzzles (£5.00 each); *Composer Companion*, a graphics utility which works with *Composer* (available from Microdeal) allowing musical data to be entered on a stave instead of Data statements (£5.00); *Hi-Text*, a hi res driver compatible with Dragon Basic, both Basic and machine code drivable (£5.00); Light Mail, a mailing list which uses a light pen and allows data entry, editing, searching, printing and label production (£5.00); *El Pea* 1 and 2, which are quite simply collections of Dragon music created on *Composer* (£3.00 each); *Help, Phone, Diary, X* (an extension of the TSC XOUT utility), and *Font* (easy access to Epson printers) all on one disc (£7.50); *Term*, which re-routes all I/O through the RS232 port to allow a terminal to be connected up (£5.00); *Curse of Comarc* (already reviewed), 47k of FLEX classic adventure (£10.00), Sherlock, a FLEX utility which allows sector editing, mapping, tracing and dumping (£10.00) and QMON, a FLEX programmers' utility which allows ASCII or hex dumps, memory searches etc. (£10.00) and (available mid December) EDT, a re configuration FLEX screen editor (price to be announced, but it won't be cheap).

More information from Microvision Software, 8 Wensley Road, Liverpool L9 8DW.



Shaolin Master from Quickbeam

Quickbeam quicker

Dave Hitchman has been in touch to tell us that as from 23rd November, Smithson Computing will become part of Quickbeam Software, ceasing to operate as a separate enterprise.

Wayne Smithson is hoping to do some programming for Quickbeam.

Dragon Monthly subscribers will be glad to hear that the newsletter will continue with Quickbeam under the new name of *Dragon's Tail*, run by Julian Cogdell, 34 Shooters Drive, Nazeing, Essex EN9 2QD. All previous subscriptions will be honoured, says Dave.

Smithson Computing's popular range which includes *Gordon Bennett* and *Electronic Author*, as well as a new game, 100 Screen Frankie, will join Quickbeam's list.

For more details contact David Hitchman, 67 Old Nazening Road, Broxbourne, Herts EN10 6RN.

Lucky thirteenth

ALAN Cook of Arc Software has been on to us to say that *The Thirteenth Task* (reviewed by Mike Gerrard in December's Adventure Trail) is now out and ready for sale and has been picking up favourable reviews in the rags. Mike liked it. The game costs £3 from Arc Software, 272 Mearns Road, Newton Mearns, Glasgow G77 5LY.

Buzz offer

Buzz Software have a number of pre-Christmas special offers, including *Small Business Telephone Directory* and *Small Business Word Processor.* For further information, send them a 17p stamp for their information pack.

Write to: Buzz Software, 15 St. Stephen Road, Penketh, Nr. Warrington, Cheshire WA5 2AN.

Quickbeam still quick

DU wishes to apologise to anyone who has waited more than 28 days for their copies of Fire Force and would like to say that any hold up has been getting the order through the Dragon User filing system, and not through any fault of Quickbeam's. All orders received before the closing date have now been dealt with.

The Answer

This is Gordon Lee's own solution to the October competiton
see page 30 for results

WE have had a number of requests for Gordon Lee's solution to his monthly competitions to be published. As there is no 'right' way to find the answer to most of his puzzles, we felt that it would be rather misleading to suggest that these were the 'proper' solutions, at the expense of someone else's equally effective program.

Many people have said that it would be a useful programming guide for them to see Gordon Lee's own answer so, we have decided to take a close look at answers all round, and accordingly are publishing here Gordon's solution to the OCTOBER competition.

Professor Hex's solution was:

```
 16532
 21536
 63251
 63512
 56321
 ------
221152
 ------
```

in which E=1, L=6, G=5, A=3 and R=2.

Solution: the program as listed computes all possible substitutions of digits for the five letters used in the sum. (Note that, with all puzzles of this type, leading zeros are not allowed, so E, L, G and R cannot be zero.)

Once this is done, it constructs the five relevant values of the words in the sum. As each digit is found its letter equivalent is added to string W$ to reconstruct the letter sequence in the answer.

These values are also stored in array W$() for reference. If all possible letter sequences are printed out (see enclosed list), the results can be compared with the two examples given. As it was stated that neither of these has any letter correctly placed, it is possible to cross off words with letters that match any letters in these key words. For example, all sequences with either an L or a G in first place can be crossed off.

This leaves just one value RREEGR, and its generating value (16532) can be obtained from array N().

```
10 CLEAR 6000
20 DIM W$(600),N(600)
30 CLS
40 N=1
50 FOR E=1 TO 9
60 FOR L=1 TO 9
70 IF L=E THEN 850
80 FOR G=1 TO 9
90 IF G=E OR G=L THEN 340
100 FOR A=0 TO 9
110 IF A=E OR A=L OR A=G THEN 330
120 FOR R=1 TO 9
130 IF R=E OR R=L OR R=G OR R=A THEN 320
140 W1=E*10000+L*1000+G*100+A*10+R
150 W2=R*10000+E*1000+G*100+A*10+L
160 W3=L*10000+A*1000+R*100+G*10+E
170 W4=L*10000+A*1000+G*100+E*10+R
180 W5=G*10000+L*1000+A*100+R*10+E
190 T=W1+W2+W3+W4+W5
200 T$=STR$(T):T$=MID$(T$,2)
210 FL=0:W$=""
220 FOR F=1 TO LEN(T$)
230 Z=VAL(MID$(T$,F,1))
240 IF Z=E OR Z=L OR Z=G OR Z=A OR Z=R
    THEN FL=FL+1
250 IF Z=E THEN W$=W$+"E"
260 IF Z=L THEN W$=W$+"L"
270 IF Z=G THEN W$=W$+"G"
280 IF Z=A THEN W$=W$+"A"
290 IF Z=R THEN W$=W$+"R"
300 NEXT
310 IF FL=LEN(T$) THEN PRINT W$;" ";
    :N(N)=W1:W$(N)=W$:N=N+1
320 NEXT R
330 NEXT A
340 NEXT G
350 NEXT L
360 NEXT E
```

Daisywheel graphics

Learn how to get squares and circles out of your printer, says ***Mike Hosken***

THERE seems to be a supposition in some quarters that only plotters and dot-matrix printers can produce printed graphics. But a daisywheel is equipped with a perfectly good dot — the full stop.

So provided the character width and line depth can be suitably adjusted there is no reason why a program cannot be devised to dump screen graphics onto paper, for example.

Among the commoner daisywheels at the lower end of the market seem to be those which are marketed as Daisystep 2000, MicroP and Quendata. Sending CHR$ codes can set the character width to anything from zero to any number of one-hundred-and-twentieths of an inch, or line depth similarly in units of forty-eighths of an inch. So to get geometric accuracy — square squares and round circles — needs five 120ths (ie, one 24th of an inch) by two 48ths (which is also one 24th of an inch). But according to the printer instruction books the codes need, for some unspecified reason, to be one greater than the number actually required, so it is code 6 for character width, and code 3 for line depth.

A screen dump can be incorporated into any graphics program. But it doesn't have to be! Having run a program and produced the last word in art or design the masterpiece will be retained in graphics memory even after NEW and loading a separate graphics dump program. But don't switch off in between, nor use PCLS, nor re-assign memory with a PCLEAR.

In pretty well its simplest form, a suitable STed here. UD is the Up/Down coordinate, LR the Left/Right. It should not be necessary to emphasise the vital importance of the punctuation in line 80. But line 100 is not essential: it simply re-initialises the printer, setting character and line dimensions back to normal as if you had switched the printer off and back on again. CHR$(12) just winds the paper right out for your added convenience.

256 characters each one-24th of an inch wide amount to something under eleven inches, fitting nicely onto A4 paper sideways. But 192 lines each one-24th of an inch deep come to just eight inches, requiring that the paper be very carefully inserted to give only a tenth of an inch top margin. Smaller versions can be produced by altering the 6 and 3 codes in line 40 — if the resulting geometrical distortion doesn't matter.

```
10 REM High resolution screen dump for daisywheel printer.
20 CLS: PRINT "CHECK THAT THE PRINTER IS READY AND THEN PRESS 'ENTER'..."
30 INPUT A$: SCREEN 1,1
40 PRINT #-2,CHR$(27)CHR$(31)CHR$(6)CHR$(27)CHR$(30)CHR$(3)
50 FOR UD=0 TO 191
60 FOR LR=0 TO 255
70 IF PPOINT(LR,UD)=0 THEN PRINT #-2," "; ELSE PRINT #-2,".";
80 NEXT LR
90 PRINT #-2: NEXT UD
100 PRINT #-2,CHR$(27)CHR$(26)CHR$(73)CHR$(12)
```



Some scientific data presented graphically using program line 40 codes of 4 and 2.

Fractal pictures

***Brian Hulley** describes the notion of fractals and tree growth*

UP until recently I was under the impression that you would have to use a mainframe to generate fractal pictures. However I soon realised that it is perfectly possible to generate fractals using the Dragon. For anyone who has escaped the flood of interest in fractals, they are simply structures which are formed when one basic shape (the initiator) can be used to derive several secondary shapes by means of a generator. For example, in figure one, a vertical line splits into two branches which then split again and again to give the image of a tree in figure four. By using the computer, we can simulate this repeated division and take the process much further than four divisions to derive a more realistic picture. In doing this the computer is not only creating a picture, it is also simulating the growth of a real tree. By then introducing a random element in the program, an even more accurate emulation of natural growth can be achieved.

The first program uses the process of repeated division to draw a tree. The formula derived in figure five is used to obtain the end coordinates of two secondary branches given the end coordinates of the primary branch and the angle this makes with the horizontal (to eliminate I/0 errors). The length of the primary branch by the branch ration, and the angle that each secondary branch makes with the primary, is the branching angle, G. (The other branch is derived by substituting -G into the equations).

Randomness is introduced in the second program, by adding two new variables, R and Z. Z contains the constant branching number and R contains the random part:

```
U=(Z+RND(R))/100
```

To modify the first program to get the second version, simply insert:

```
145 U=(Z+RND(R))/100
445 INPUT "RANDOM OFFSET (0) ";R
```

Then change:

```
80 Y(1)=49:G=0.43633:R=0:Z=80
440 ?:INPUT "BRANCHING NUMBER (80) ";Z
```

Also change "VERSION 1" to "VERSION 2" in line 40 and "UNIFORM" to "UNIFORM/RANDOM" in line 50.

When you run this program without entering new values you will get a uniform tree as before. To introduce the randomness, enter 40 for the branching number, and 65 for the random offset. Of course, a little experimentation is required to get the best values for a realistic tree.

There is still quite a lot that could be added to this program. For example, the branch angle could be made random, or a random factor could decide whether two secondary branches or one would be formed each time. Alternatively, a small routine could be added at the beginning of the program to draw several trees, each with its own branching angle, branching number .... and so on.

However, fractals can also be used to good effect in the creation of snowflakes. This time, we begin with an equilateral triangle, then produce an equilateral 'kink' on each side, then produce kinks on each side of the first kink as in figures six to eight. The program takes the process to the limit of the Dragon's resolution. Unlike the tree program, there is no necessity for a random element to be present, since snowflakes are perfectly symmetrical. Also, this program is slightly more complicated since more variables have to be dealt with each time, although it behaves in a very similar way. The formula used is derived in figure nine. One of the main problems is that the program must know which way up to put the kink. The routine at 320 deals with this by looking at which way round the coordinates have been sent to it, then swapping the variables back the right way round and setting a flag, which is later used to determine

Figure 1



Figure 2



Figure 3

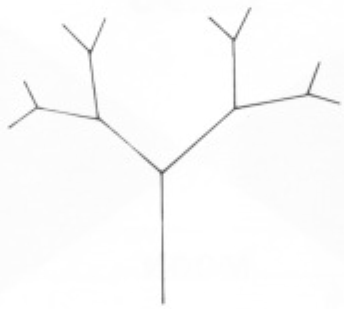


Figure 4

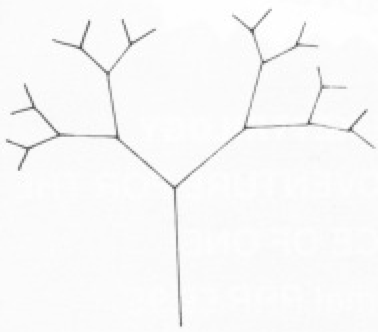
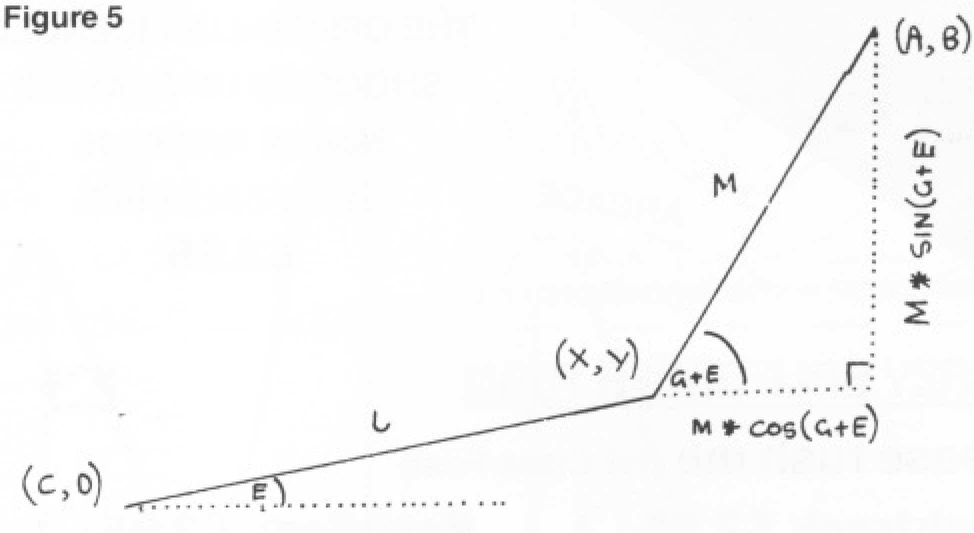


Figure 5



so M = U*L = U*SQR((-C)*(X-C)+(Y-D)*(Y-D))
and A = X + M*COS(G+E)
B = Y + M*SIN(G+E)

Notice: U is the branching ratio.

the correct angle to use (see figures ten and eleven). The other main problem is that the Dragon has no ARCCOS command. The correct formula for the inverse cosine is:

1.57-atn(X/sqr(-X*X+1))

You can use this program to generate transverse sections of icebergs or crystals by changing the coordinates in line 80, ie so that a scalene triangle is used. The program can also be modified so that a pentagon is used instead of an equilateral triangle. To do this, simply alter line 70 so that all arrays are dimensioned with 320 instead of 192; replace 3 by 5 in lines 140,160,210 (where 3*4 ↑V-1 occurs); change the coordinates in line 80 setting X(0 to 5) and Y(0 to 5) with the required values, and change 100 to 120 to draw five lines instead of three.

The tree and the snowflake crystal are by no means the only structures which are composed of fractals. The coastline of Britain is perhaps the best known example of a fractal, and in fact it was in trying to answer the question "How long is the coastline of Britain?" that led Mandelbrot to formulate his theory of fractals in 1975. The Dragon can be used to simulate the development of all sorts of natural structures, from continents to microscopic plants.

More information on fractals can be obtained from *The Fractal Geometry of Nature* by Mandelbrot, published in 1982.

Figure 6

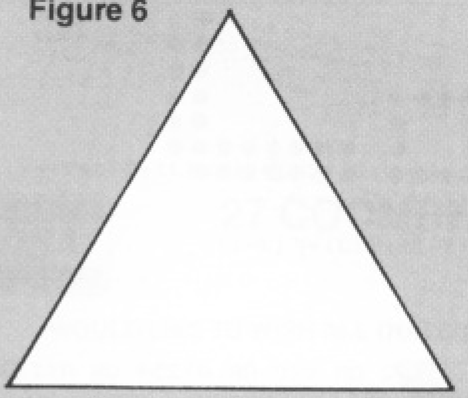


Figure 7

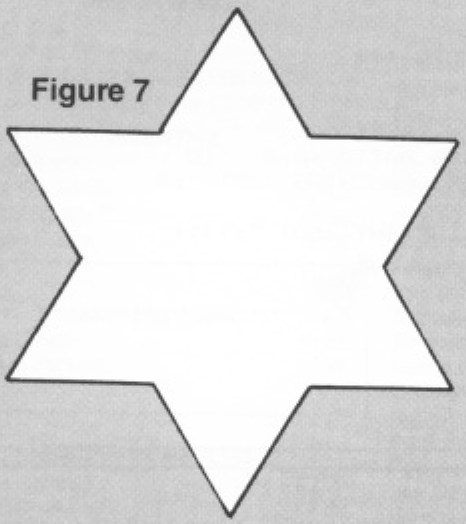


Figure 8

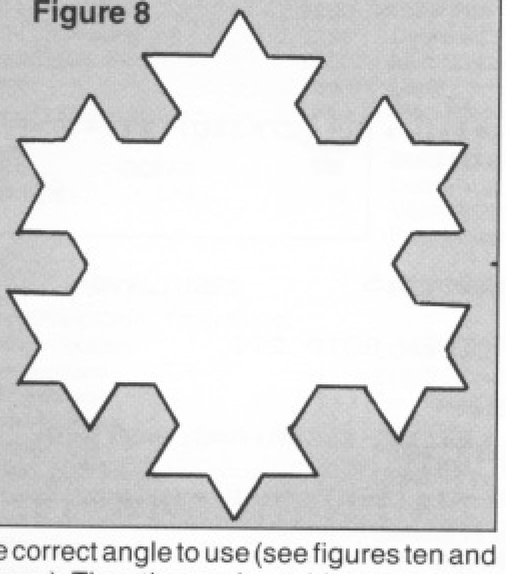
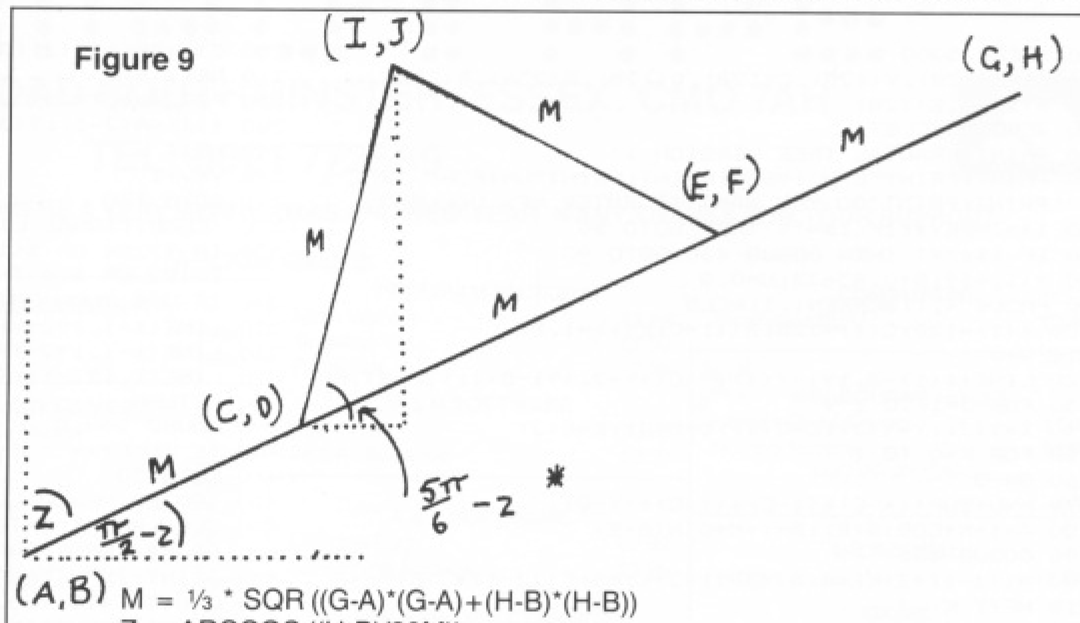
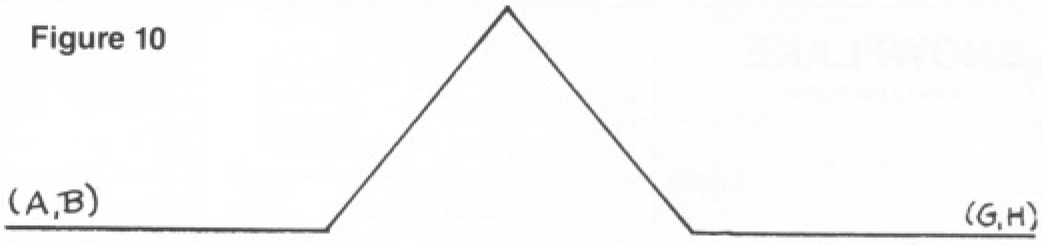


Figure 9



M = ⅓ * SQR ((G-A)*(G-A)+(H-B)*(H-B))
Z = ARCCOS ((H-B)/30M))
N = $\frac{5\pi}{6}$ -Z or $\frac{\pi}{6}$ -Z depending on whether the kink is to be facing upwards or downwards
I = M*COS(N)+C
J = M*SIN(N)+D
C = ⅓ * (G+2*A)   D = ⅓*(H+2*B)   E = ⅓*(2*G+A)   F = ⅓*(2*h+B)

Figure 10



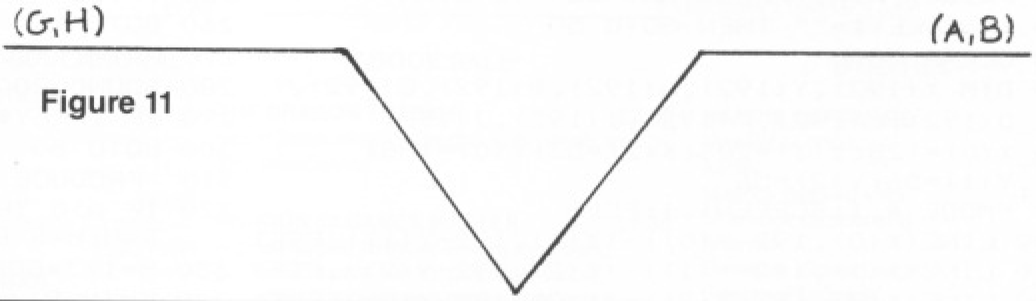


Figure 11

Program notes

Tree

10-1000 Initialise variables, input parameters, clear screen etc.
110 Initialise level counter.
120 Draw the 'trunk' of the tree.
130-220 Add branches onto the last primary branches stored in X(J),Y(J) and store the coordinates of the secondary branches in A(),B() and the new angle in F().
230 If level = 7 then the arrays are full, so wait until a key is pressed before starting again.
240-260 Transfer arrays X(),Y() into C(),D().
270-290 Transfer arrays A(),B() into X(),Y().
300-380 Increment level counter then loop back to a further recursive division.
320-380 Routine to draw a line of variable thickness depending on the level. It does not draw the line if any of the coordinates are off the screen.
390-410 Wait routine.
420-490 Routine to get new parameters.

Snowflake

10=70 Intro + dimension arrays.
80 Initialise variables.
90-120 Draw triangle on screen.
130 Initialise level counter.
140-190 go through each line in previous level, producing kinks and storing new variables in A(),B(),C(),D(),E(),F(),I(),J().
200 Goto routine and wait.
210-240 Get arrays set up for next time round transferring A()......into X(......),Y(....).
250-260 Increment level counter then loop back.
270-300 Paint snowflake then wait for keypress etc.
320- Routine to produce a kink the right way up as in figures ten and eleven.

In both programs V is used as a level counter. The other variables are as shown in the formula diagrams, five and nine.

TREE

```
10 CLEAR 5000
20 DIMX(128),Y(128),C(128),D(128),A(256),B(256),
   F(256),E(128)
30 SCREENO:CLS7
40 PRINT"FRACTAL TREE VERSION 1"
50 PRINT:PRINT"B/H 1985":PRINT:PRINT"UNIFORM TREE":
   PRINT:PRINT"DO YOU WANT TO ENTER NEW VALUES?";
60 I$=INKEY$:IF I$="" THEN GOTO 60
70 IF I$="Y" THEN GOSUB 430:GOTO 90
80 Y(1)=49:G=0.43633:U=0.8
90 PMODE 4,1:SCREEN1,1:PCLS
100 X(1)=128:C(1)=128:D(1)=0:E(1)=1.57
110 V=0
120 LINE(X(1)-2,191-Y(1))-(C(1)+2,191-D(1)),PSET,BF
130 FOR J=1 TO 2^V
140 X=X(J):Y=Y(J):C=C(J):D=D(J):E=E(J)
150 FOR K=0 TO 1
160 G=-G
170 M=U*SQR((X-C)*(X-C)+(Y-D)*(Y-D))
180 A=X+M*COS(G+E):B=Y+M*SIN(G+E)
190 GOSUB 330
200 A((J-1)*2+K)=A:B((J-1)*2+K)=B:F((J-1)*2+K)=E+G
210 NEXT K
220 NEXT J
230 IF V=7 THEN GOTO 390
240 FOR J=1 TO 2^V
250 C(J*2-1)=X(J):D(J*2-1)=Y(J):C(J*2)=X(J):D(J*2)=Y(J)
260 NEXT J
270 FOR J=1 TO (2^V)*2
280 X(J)=A(J-1):Y(J)=B(J-1):E(J)=F(J-1)
290 NEXT J
300 V=V+1
310 GOTO 130
320 'CONDITIONAL LINE
330 IF X>254 OR X<1 OR Y>192 OR Y<0 OR A>254 OR A<1 OR
    B>192 OR B<0 THEN GOTO 380
340 IF V>5 THEN GOTO 370 ELSE IF V>2 THEN GOTO 360
350 LINE(X+1,192-Y)-(A+1,192-B),PSET
360 LINE(X-1,192-Y)-(A-1,192-B),PSET
370 LINE(X,192-Y)-(A,192-B),PSET
380 RETURN
390 SOUND 200,5
400 IF INKEY$="" THEN GOTO 400
410 GOTO 30
420 'ENTER NEW VALUES
430 CLS6:PRINT"ENTER NEW VALUES"
440 PRINT:INPUT"BRANCHING RATIO (0.8) ";U
450 PRINT:INPUT"ANGLE OF BRANCHING (25) ";G
460 G=G*3.1415/180
470 PRINT:INPUT"INITIAL HEIGHT (50) ";Y(1)
480 Y(1)=Y(1)-1
490 RETURN
```

SNOWFLAKE

```
10 CLS7
20 PRINT"FRACTAL SNOWFLAKE"
30 PRINT:PRINT"B/H 1985"
40 PRINT:PRINT"PRESS ANY KEY TO START"
50 IF INKEY$="" THEN GOTO 50 .
60 CLEAR 5000
70 DIM X(192),Y(192),A(192),B(192),C(192),
   D(192),E(192),F(192),I(192),J(192)
80 X(0)=128:X(1)=203:X(2)=53:Y(0)=186:
   Y(1)=56:Y(2)=56
90 PMODE 4,1:SCREEN1,1:PCLS
100 LINE(X(0),192-Y(0))-(X(1),192-Y(1)),PSET
110 LINE(X(1),192-Y(1))-(X(2),192-Y(2)),PSET
120 LINE(X(2),192-Y(2))-(X(0),192-Y(0)),PSET
130 V=0
140 FOR T=0 TO 3*4^V-1
150 A=X(T):B=Y(T)
160 IF T=3*4^V-1 THEN G=X(0):H=Y(0)
    ELSE G=X(T+1):H=Y(T+1)
170 GOSUB 320
180 A(T)=A:B(T)=B:C(T)=C:D(T)=D:E(T)
    =E:F(T)=F:I(T)=I:J(T)=J
190 NEXT T
200 IF V=3 THEN GOTO 270
210 FOR O=0 TO 3*4^V-1
220 T=O*4
230 X(T)=A(O):Y(T)=B(O):X(T+1)=C(O):Y(T+1)
    =D(O):X(T+2)=I(O):Y(T+2)=J(O):X(T+3)=E
    (O):Y(T+3)=F(O)
240 NEXT O
250 V=V+1
260 GOTO 140
270 PAINT (128,128),5,5
280 SOUND 200,5
290 IF INKEY$="" THEN GOTO 290
300 GOTO 80
310 'PRODUCE "KINK"
320 IF A>G THEN LET U=-1:S=A:A=G:G=S:S=B:
    B=H:H=S ELSE U=1
330 M=1/3*SQR((G-A)*(G-A)+(H-B)*(H-B))
340 K=(H-B)/(M*3)
350 Z=1.57-ATN(K/SQR(-K*K+1))
360 C=1/3*(G+2*A):D=1/3*(H+2*B)
    :E=1/3*(2*G+A):F=1/3*(2*H+B)
370 IF U=-1 THEN N=0.5236-Z ELSE N=2.6179-Z
380 I=M*COS(N)+C:J=M*SIN(N)+D
390 LINE(A,192-B)-(G,192-H),PRESET
400 LINE(A,192-B)-(C,192-D),PSET
410 LINE(C,192-D)-(I,192-J),PSET
420 LINE(I,192-J)-(E,192-F),PSET
430 LINE(E,192-F)-(G,192-H),PSET
440 IF U=-1 THEN S=A:A=G:G=S:S=B:B=H:H=S:
    S=C:C=E:E=S:S=D:D=F:F=S
450 RETURN
```

It takes two to Kung-Fu

Program: *Shaolin Master*
Supplier: Quickbeam Software
Price: £8.45

THE SECOND of Quickbeam's games reviewed this issue is a version of the two-player Kung-Fu games so often found in pubs, amusement arcades, etc. The game can be played by one player against a rather deadly computer opponent or two players can play each other (which gives this game one good point already).

The game is in two sections, there's the duel between you and an opponant, and then there's the rather gimmicky corridor of death in which you run down a corridor kicking and jumping various large spiky objects.

The game rather like real Kung-Fu needs practice before one can become deadly at it. The controls are a nightmare to start with but, like driving a car, after a while they become natural, even second nature. However, to give you an idea of what you're letting yourself in for, the game has the ability to jump up, high punch, move right, low punch in kneeling position, duck, back somersault, move left, forward somersault, stand up, drop kick, high kick, medium kick, low lick, forward leg sweep, back leg sweep, turn around, high back kick, and low back kick, and all these operations are performed with ONE joystick!

The graphics are as excellent as those of *6809 Express* (and that's pretty good) but the game is about thirty times as hard, and thirty times as interesting.

Now for some nit picking. The game comes in an attractive cassette box but the instructions are on a separate printed leaflet. Although this seems like a good idea at the time, there should be a reprint of the joystick configurations on the inside of the case along with what each can do, as I managed to lose my sheet three days after receiving the game and thus rendered the game inoperative until I found it two months later!

Also, the collision detection routine needs some work on it as it is all too easy for you to see your figure kick the other guy sxquarely in the groin only to find that you've actually done no damage to him whatsoever! And then, to add insult to injury, the demon computer assassin will knock you flat with a flying axe kick three inches away from you! I exaggerate, of course, but please, and this goes for all programmers, remember that all we have to judge by is what we can see on the screen!

To sum up then, a really good game. Let's hope *Fire Force* is just as good, because it will surely establish Quickbeam - who now have Wayne Smithson working for them, and he is a good programmer — as one of the market leaders.

Jason Orbaum

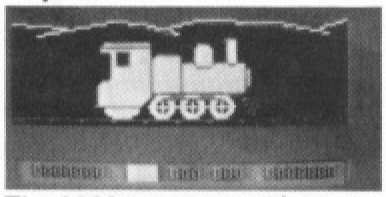
Trouble on the 6809 Express

Program: *6809 Express*
Supplier: Quickbeam Software
Price: £6.45

OOOOOOOOOOOH!

The runaway game came over the hill and she blew.
The runaway game came over the hill and she blew.

The runaway game came over the hill
And has spent several months on my window sill
'Cause it's too, too, too boring for you.



The 6809 comes past the trees and it blows.
You play a Nazi but that's just the way the game goes.

The programminge's all been disigned to please
But there's trouble with the ideologies
Unless you, you, you are not a Jew.

The 6809 is a split screen game that goes fast
Below is the top view above are the things as they're passed.

The graphics are great and the sound is hot
It's just a great shame that the game is not.
'Cause it's too, too, too bad to be true.

Controlling the game requires just as much skill as sleep.
And I think that I've had more fun having a chat with a sheep.

And just 'cause you know how to draw on the screen
Doesn't mean that you've then got a game that's supreme.
So I'm sorry Dave, but this one just won't do.

Jason Orbaum

Turtle graphics

Program: *LOGO 3.4*
Supplier: Bernd Knechtel Software, etc.
Price: tba

THE latest piece of DragonDOS software to arrive from Bernd Knechtel is a utility which interfaces directly with the Dragon's own fair Basic and provides a few useful additions to the already excellent graphics commands.

LOGO 3.4 provides the user with four extra graphics commands which allow turtle-type graphics to be programmed. Upon running a program an invisible 'turtle' is placed at the centre of the screen facing upwards (zero degrees) and this turtle may be manipulated very easily with the four commands mentioned above. For those who have not had the pleasure of using turtle graphics before, this involves 'guiding' a fictional turtle around the graphics screen with the option of leaving, or not leaving, a line drawn in its wake. So it is with a turtle. The first two commands in LOGO 3.4 are TRIGHT and TLEFT which allow you to turn the turtle left or right through some specified angle to point in the direction you wish to draw. The position you wish to start from may be defined and redefined using the TSTART command which sets the initial position of the turtle on the screen. And the line may be drawn (or the turtle moved) using the TMOVE command. The TMOVE command which specifies the distance the thrtle is to move (in pixels) has an optional parameter which allows a line to be drawn without moving the turtle, or moving the turtle without drawing a line.

Using the example given earlier in the text, the following program would instruct the turtle to draw a square on the screen.

```
10 MODE4:PCLS:SCREEN 1,1
   — Set up graphics mode
20 TSTART 128,96
   — Move to centre of the screen
30 TMOVE 50
   — Draw a line 50 pixels long
40 TRIGHT 90
   — Turn through 90 degrees
50 TMOVE 50
   — Draw a line 50 pixels long
60 TRIGHT 90
   — Turn through 90 degrees
70 TMOVE 50
   — Draw a line 50 pixels long
80 TRIGHT 90
   — Turn through 90 degrees
90 TMOVE 50
   — Draw a line 50 pixels long
100 END
```

This is obviously a very tedious way of achieving such a simple objective and so, because LOGO interfaces directly with Dragon Basic, a loop may be written which obviously allows far greater flexibility.

For someone who uses Dragon Basic a lot to draw graphics, LOGO 3.4 is a very useful addition to the programmer's armoury. A full assembler listing of LOGO is supplied with the program for anyone daft enough to try and modify it and the package carries no anti-copying devices which makes it easy to incorporate into your own programs. Although LOGO only adds four simple commands to Basic I can't think of anything else that could be added. LOGO does give the benefits of turtle graphics without the usual pain of having to learn an entirely new environment.

Roy Coates

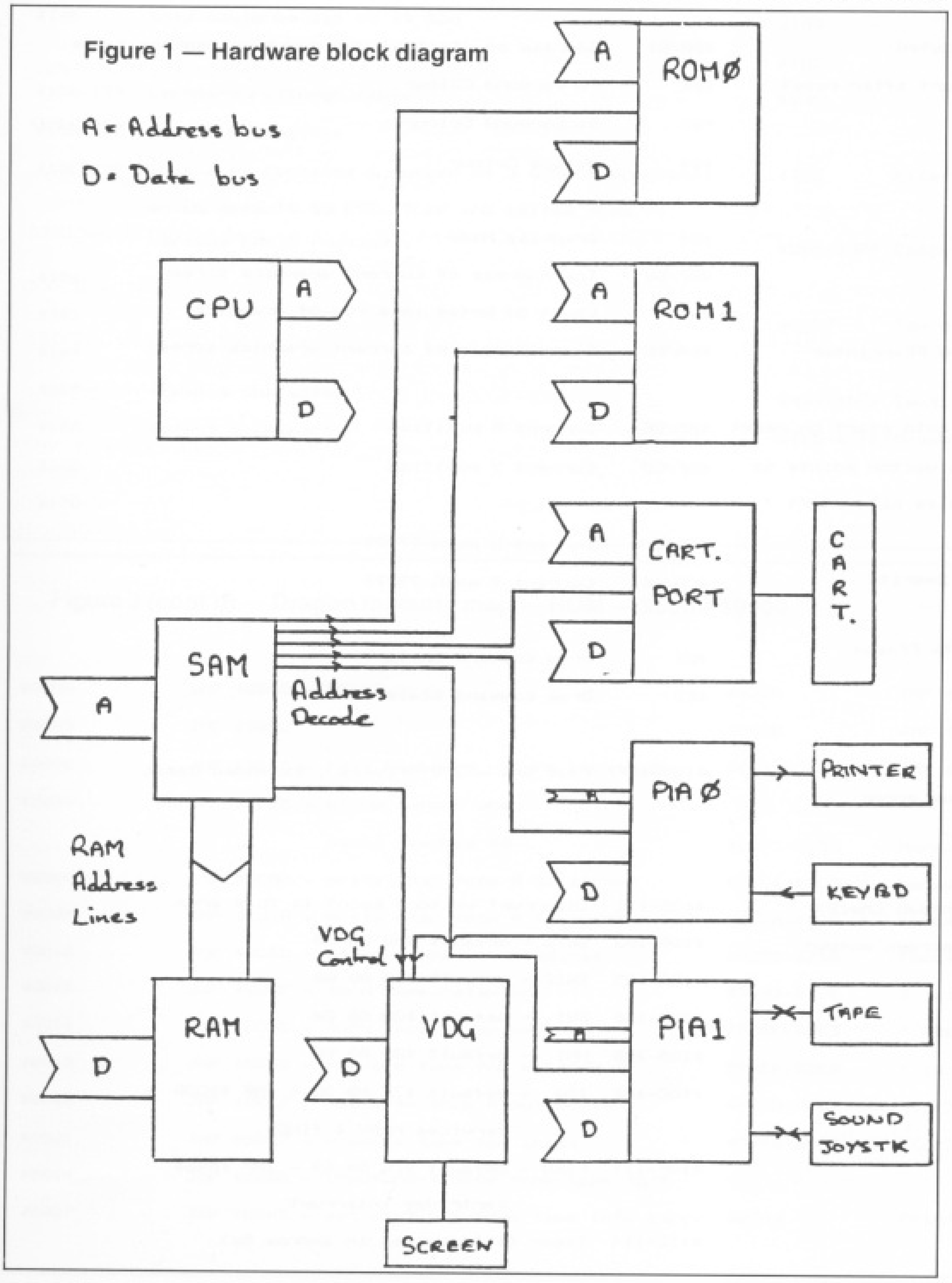
Inside the 32

Dave Barnish strikes a light on the Dragon's memory map

THE average user of the Dragon 32 computer probably sits quite happily at his or her computer, turns it on and starts programming in Basic or using applications programs (or games) written by somebody else, without a lot of concern about how the Dragon really works. Anyone who has dabbled in machine code may sometimes wonder at the complexity of a system that can take a microprocessor, which only understands binary numbers, and produce a computer capable of interacting with the outside world via a keyboard and screen in a language which is not too far removed from English. This article is intended to take a little of the mystery out of the subject and may help those of you feeling confident enough to modify the basic operation of the machine by enhancing the Basic or to use Basic routines in your own machine code programs.

At the heart of the Dragon is a 6809 microprocessor which is the computer's central processing unit (CPU), where the brainwork goes on. In order for the CPU to do anything it must have instructions, and these are stored in Read Only Memory (ROM). The instructions are built into the ROM when it is made and cnnot be altered; they are retained even when the computer is switched off. Random Access Memory (RAM) on the other hand provides a temporary store of data which may be changed at will but is lost when the power is off. In order to communicate with the outside world, some facility for input/output (I/O) is required. On the Dragon, the I/O is provided by two general purpose I/O chips (peripheral interface adapters or PIAs) which, with their associated electronics, provide the keyboard, cassette, sound, printer and joystick interfaces. The screen is a special case of I/O and is handled by a special video display generator chip (VDG).

Figure 1 — Hardware block diagram



Because the 6809 is an 8 bit processor, all data is handled in groups of eight bits (BInary digITS) called bytes. For this reason RAM, ROM and I/O are split into a number of locations, each of which can hold one byte, and in order that the CPU may distinguish one location from another, each is given an individual address which is a number in the range 0 to 65535. The address decoding which organises what goes where is contained in a large chip called a synchronous address multiplexer (SAM), which also does a bit of display management, and maintains the contents of RAM amongst other things. A block diagram of the Dragon's hardware is given in Figure 1, which shows the interrelation between each device. The 6809/SAM and VDG combination is a standard chip set produced by Motorola which has also been adopted by the Tandy colour computer. This and the common origin of their Basics explains their internal similarities.

Since the addressing is not based on the decimal system but on binary, it seems sensible not to use decimal ourselves when trying to understand what's going on. However, binary is very unwieldy, so generally the hexadecimal (base 16) system of counting is used which makes things much more managable. Each hexadecimal (hex) digit has a range of 0 to 15 and to get the numbers 10-15 into one digit, the characters A-F are used. Hexadecimal numbers are in general preceded by a $ sign, which is a much more conventional way of indicating hex than the &H that Dragon Basic uses. Figure 2 gives an example of various numbers expressed in decimal, binary and hexadecimal. Note that each hex digit refers directly to a group of four binary digits (four binary digits make up half a byte which is known as a nibble).

Figure two — Examples of decimal, binary and hexadecimal numbers

Decimal	Binary	Hexadecimal
1	1	1
2	10	2
9	1001	9
10	1010	A
11	1011	B
15	1111	F
16	10000	10
19	10011	13
20	10100	14
31	11111	1F
32	100000	20

A list of which location is assigned to which physical device is known as a memory map, an example of which can be seen on the back page of the Dragon's user manual. This map gives a general idea of where things are, but for a serious investigation of how the system works a more details map is required. Such a map is provided in Figure 3 and the rest of this article should be read in conjunction with this map.

In order to appreciate how the Dragon works we had best start at the beginning and find out what happens when the computer is first switched on. When power is first applied to the machine its electronics come on in an undertermined state. In order to set everything to a known configuration the CPU, I/O and SAM chips are providced with a reset input line which is held low (grounded), for a short time after power up by a simple circuit. When the reset line is low, the PIAs are cleared and the SAM is set to its initial configuration enabling the address decoding and locking in to the VDG's timing. The CPU then looks at addresses $FFFE and $FFFF which SAM remaps to ROM locations $BFFE and $BFFF. Here it finds the reset vector which is the address of the reset routine to which control is passed. This routine initialises all the I/O, sets up the Basic system variables to their default values, and clears the screen. Having got this far the first two locations of the cartridge memory area are checked ($C000 and $C001) and if they contain $44 and $4B respectively then control is passed to the cartridge software at locations $c002. If they two bytes are not set as described the reset routine prints the startup message from locations £B4B3 onwards and passes control to the main Basic operating systems.

Figure 3 — Dragon memory map — RAM

$19/1A	pointer to beggining of BASIC text
$1B/1C	pointer to beggining of variable space
$1D/1E	pointer to beggining of array pointer table
$1F/20	(initialised to beginning of variable space)
$21/22	Top of stack (stack grows down)
$23/24	Top of string free space
$27/28	Highest Ram available to BASIC
$2D/2E	pointer to statement to be executed
$2F/30	Text pointer for BASIC warm start after reset
$31/32	DATA line number
$33/34	DATA pointer
$35/36	INPUT pointer
$68/69	Current Line Number
$6F	Current Device 0=screen FF=tape FE=printer
$70	End of file flag 0=char found
$71	Restart flag - if <> $55 then cold start on reset
$72/73	Restart Vector - if flag=$55 & vector points to a NOP then warm start to NOP else cold start
$78	tape status - 0=closed 1=read 2=write
$7C	tape block type 0=header 1=data FF=eof
$7D	No. of bytes to tape
$7E/7F	Base address of bytes to tape
$81	Tape error code 0=ok 1=checksum error 2=memory error
$87	Last key pressed (cleared by break check)
$88/89	pointer to next location for screen output
$8C	Pitch for PLAY
$8D/8E	Duration in PLAY
$8F	Cursor Blink Count
$90/91	Tape Leader Byte Count
$95/96	Tape motor delay

$99	Printer comma field width
$9A	Last comma field width
$9B	Printer width
$9C	Printer head position
$9D/9E	Exec Address
$9F-AA	Start of self modifying routine continued in rom reads a char from BASIC text.
	$A6/A7 - text pointer to current active byte
$B0/B1	Address of start of USR function address table
$B2	Foreground Colour
$B3	Background Colour
$B4	Active Colour
$B6	Graphics Mode
$B7/B8	Top Address of current graphics screen
$B9	Count of bytes in a row of graphics
$BA/BB	Base address of current graphics screen
$BD/BE	Current X position
$BF/C0	Current Y position
$C8/C9	Current X posn ?????
$CA/CB	Current Y posn ?????
$E8	Draw command Angle
$E9	Draw command Scale
$100-$1FF	Page One - I/O Drivers , Extended Basic
$100-111	Interrupt vectors point to this area
$100-102	SWI3 - default $00 00 00
$103-105	SWI2 - default $00 00 00
$106-108	SWI - default $00 00 00
$109-10B	NMI - default $00 00 00
$10C-10E	IRQ - default $7E 9D 3D - JMP $9D3D services PLAY & TIMER
$10F-111	FIRQ - default $7E B4 69 - JMP $B469 cartridge interrupt
$112-114	Timer ($114 cycles in approx 5s)

$120	STUB0 used in setup of BASIC keyword access
	No of normal reserved words
$121/122	Address of normal reserved word list
$123/124	Address of statement despatch table
$125	No of function reserved words
$126/127	Address of function reserved word list
$128/129	Address of function despatch table
$12A-133	STUB1 dummy to make end of stubs. Layout as STUB0
	except $12D - address stat desp routine
	$132 - address func desp routine
$134-13D	STUB2 as STUB1
$134-147	Normally used for USR function address.
$148	Printer Buffer full auto LF flag default=FF (on)
	if 0 then CR printed at EOL
$149	Alpha lock flag default=FF=locked 0=unlocked
$14A	Printer end of line sequence.
	No. of chars to print at EOL (def=1)
$14B	default = $0D (return)
$14C	default = $0A (line feed)
$14D/14E	default = $00 (null)
$14F	this could be bit of it too
$150-159	keyboard rollover table
$15A-15D	Joystick readings
$15E-1AF	This area contains a number of 3 byte subroutines
	which default to RTS. They are called from
	various BASIC routines.
$15e	
$161	
$164	
$167	Input a character
$16A	Output a character
$16D	
$170	
$173	
$176	
$179	
$17C	
$17F	
$182	Read an input line
$185	
$188	
$18B	Evaluate an expression
$18E	User error trap
$191	System error trap
$194	RUN
$197	NEW
$19A	Read in next statement
	After RTS break is checked
	so can be used to disable break.
$19D	
$1A0	
$1A3	Crunch BASIC line for storing
$1A6	Decrunch BASIC line for output
$1A9	
$1AC	
$1AF	
$1E2	File type ?? 0=BASIC 1=data 2=machine code
$200-$3FF	Cassette Buffers etc.
$03D7	Top of hardware stack (stack goes down)
$400-$5FF	Text screen Default area
$600-$7FFF	Graphics Screens , BASIC user ram

Figure 3 (cont'd) — Dragon memory map — ROM — $8000-$9000

$8000	JMP $BB40 - RESET
$8003	JMP $BB88
$8006	JMP $BBE5 - Poll keyboard return key in A
$8009	JMP $BBB5 - blink cursor when count reaches 00
$800C	JMP $BCAB - write char from A to screen
$800F	JMP $BD1A - write char from A to printer
$8012	JMP $BD52 - Update joystick readings
$8015	JMP $BDCF - turn tape relay on
$8018	JMP $BDDC - turn tape relay off
$801B	JMP $BE68 - prepare tape for writing
$801E	JMP $BE12 - put byte from A out to tape
$8021	JMP $BDE7 - prepare tape for input
$8024	JMP $BDAD - input next byte from tape to A
$8027	JMP $BDA5 - get next bit from tape into carry
$802A	JMP $BE7B
$802D	JMP $BE7C
$8030	JMP $BE7D
$8033-8153	Normal reserved word table
$8154-81C9	Normal despatch table
$81CA-824F	Function reserved word table
$8250-8293	Function despatch table
$82A9-82DE	2 letter error messages
$82DF-82E5	' ERROR' in ascii
$82E6-82EA	' IN ' in ascii
$82EB-82EF	'<cr> OK<cr>' in ascii
$82F0-82F6	'<cr> BREAK ' in ascii
$8344	Print error message - error no *2 in B

$8415-8447	NEW	$8E96	PEEK
$8448-	FOR	$8E9D	POKE
$8514	RESTORE	$8EA4-	LLIST
$8532-	END	$8EAA	LIST
$8539-	STOP	$8F08	Decrunch tokenised BASIC to text
$8560	CONT	$8F67	Crunch text to tokenised BASIC
$8571	CLEAR	$903D-	PRINT
$85A5	RUN	$923C	LOG
$85B9-	GO	$9425	SGN
$85F3	RETURN	$943E	ABS
$8613	DATA	$9499	INT
$8616-	REM and ELSE	$957A	print 2 byte unsigned number from D
$8647-	IF		
$8675	ON	$9697	SQR
		$9713	EXP
		$9772	RND
$86BC	LET	$97CB	COS
$872B-	INPUT	$97D1	SIN
$8777	READ	$9816	TAN
$8829-	NEXT	$9877	ATN
$8A8B-	DIM	$9956	FIX
$8B30	Internal conversion routine	$9965	EDIT
	passes BASIC parameters to machine code	$9AD9	TRON
$8C31	MEM	$9ADA	TROFF
$8C39	pass integer in D to BASIC	$9ADE	POS
$8C40	STR$	$9AF4	VARPTR
$8DC7	LEN	$9B84	STRING$
$8DD2	CHR$	$9BB4	INSTR
$8DE6	ASC	$9C81-	DEF
$8DF1	LEFT$	$9D1D	USR
$8E0E	RIGHT$	$9D3D	IRQ routine - TIMER and PLAY serviced here
$8E15	MID$	$9D59	TIMER
$8E5C	VAL	$9D61	DEL
$8E83	Get number from BASIC into X	$9DFA	RENUM

Figure 3 (cont'd) — Dragon memory map — ROM — $A000-$BFFF

$A000-A00D	indirect jump vectors		
$A000	$8006 - poll keyboard	$A9AF	FMODE
$A002	$B54A	$A9FE	SCREEN
$A004	$8021 - prepare tape for input	$AA19	PCLEAR
$A006	B93E	$AABE	PCOPY
$A008	$B999	$AAF0	GET
$A00A	$8012 - update joysticks	$AAF3	PUT
$A00C	$801B - prepare tape for writing	$AC87	PAINT
		$ADBD	PLAY
$A00E	HEX$	$B051	DRAW
$A049	DLOAD		
$A6C7	PPOINT	$B238	CIRCLE
$A6EF	PSET	$B3B4	RESET routine
$A6F3	PRESET	$B469	FIRQ routine - cartridge interrupt
$A749	LINE	$B4B3-B504	Startup message:-
$A8C0	PCLS		(C) 1982 DRAGON DATA LTD <cr>
$A8D4	COLOR		16K BASIC INTERPRETER 1.0 <cr>

	(C) 1982 BY MICROSOFT <cr><cr> $00
$B54A	print A to current device
$B64C	CLOSE
$B682-	CSAVE
$B6D4-	CLOAD
$B770	EXEC
$B796	INKEY$
$B800	EOF
$B81E	SKIPF
$B828	OPEN
$B981	MOTOR
$B9D2-	SET
$BA03	RESET
$BA44	POINT
$BA5F	CLS
$BA77	clear text screen
$BA9A	SOUND
$BAA5	play one note
$BADF	AUDIO
$BB0D	JOYSTK

$BE7F-BFF1	unused area - set to $00s
$BFF2-BFFF	This area contains the reset and interrupt vectors. The rom is re-mapped by SAM to appear at $FFF2-FFFF which is where the 6809 expects these vectors to be.
$BFF2-BFF3	SWI3 vector - $0100
$BFF4-BFF5	SWI2 vector - $0103
$BFF6-BFF7	FIRQ vector - $010F
$BFF8-BFF9	IRQ vector - $010C
$BFFA-BFFB	SWI vector - $0106
$BFFC-BFFD	NMI vector - $0109
$BFFE-BFFF	RESET vector - $B3B4

As well as the reset vector the area $BFF2 to $BFFF contains various other vectors which all point to an area of RAM at $100 and $111. Only two of these vectors are used by the system; $BFF6 and $BFF7 which hold the IRQ (Interrupt ReQuest) vector. These vectors point to locations $10C and $10F respectively, which hold instructions causing a jump to the appropriate interrupt service routine. In the case of the IRQ, which is called fifty times a second by the video refresh signal, the service routine looks after the TIMER function and the PLAY command while the FIRQ, which is called on power up if a cartridge is present, passes control to software help in the cartridge. There are four other vectors present which are not used by the system and point to RAM areas which contain $00. These are the SW1, SW12 and SW13 (Software Interrupt) vectors, which are called when the CPU executes the instructions SW1, SW12 and SW13 respectively, and the NMI (Non-Maskable Interrupt) vector which is called when the NMI line to the CPU is held low. By writing your own service routine and inserting a jump to it in the appropriate locations ($100-$111) you can use these interrupts for your own purposes, eg if you have a piece of hardware which requires fast attention from the CPU occasionally, such as a disc drive, you could have your hardware generate an NMI and call your service routine. The machine code monitor 'Encoder 09' by Premier uses the SW1 vector to prodive breakpoints in user programs. When a SW1 is encountered Encoder's breakpoint routine is called and there is an option to continue the user program, ie return from the interrupt.

Perhaps more useful is the fact that you can redirect the system interrupts to do your own thing, eg the IRQ interrupt could be made to execute a routine for displaying the time on the screen before jumping back to

Figure 4 Interrupt driven real time clock

```
7869                ********************************************
7869                *                                          *
7869                *Figure 4 - Interrupt driven real time clock*
7869                *                                          *
7869                *Clock is started using EXEC&H7800         *
7869                *                                          *
7869                *Set time as follows :-                    *
7869                *POKE&H7862,&H(seconds)                    *
7869                *POKE&H7863,&H(minutes)                    *
7869                *POKE&H7864,&H( hours )                    *
7869                *                                          *
7869                ********************************************
7869                *
7869                *Change IRQ vector to point to the clock routine
7869                *
7800                        ORG     $7800
7800   CC 7807      INIT    LDD     #NEWIRQ
7803   FD 010D              STD     $010D
7806   39                   RTS
7807                *
7807                *Main routine
7807                *
7807   8E 7861      NEWIRQ  LDX     #CLOCK
780A 108E 7865              LDY     #CHECK
780E                *
780E   A6 84        NXTDIG  LDA     ,X        get clock digit
7810   8B 01                ADDA    #$01      increment digit
7812   19                   DAA
7813   A7 84                STA     ,X
7815   A1 A0                CMPA    ,Y+       check for overflow
7817   26 07                BNE     NOINC
7819   6F 80                CLR     ,X+       zero digit
781B   8C 7865              CMPX    #CLOCK+4 check for hours overflow
781E   26 EE                BNE     NXTDIG    do next digit
7820                *
7820                *Put clock on screen
7820                *
7820 109E 88        NOINC   LDY     $88       save old screen position
7823   CC 0417              LDD     #$0417    point to clock position
7826   DD 88                STD     $88
7828                *
7828   8E 7865              LDX     #CLOCK+4
782B   A6 82        NXTCHR  LDA     ,-X       put characters on screen
782D   BD 784A              JSR     OUT
7830   86 3A                LDA     #$3A
7832   BD BCAB              JSR     $BCAB     put char in acc A on screen
7835   A6 82                LDA     ,-X
7837   BD 784A              JSR     OUT
783A   86 3A                LDA     #$3A
783C   BD BCAB              JSR     $BCAB
783F   A6 82                LDA     ,-X
7841   BD 784A              JSR     OUT
7844                *
7844 109F 88                STY     $88       restore old screen pointer
7847   7E 9D3D              JMP     $9D3D     do normal IRQ routine
784A                *
784A                *
784A                *
784A                *
784A                *
784A                *
784A                *
784A                *
784A                *
784A                *
784A                *This subroutine puts one digit on the screen
784A                *
784A   1F 89        OUT     TFR     A,B       convert from packed
784C   84 F0                ANDA    #$F0      binary coded decimal
784E   47                   ASRA              to two ASCII characters
784F   47                   ASRA
7850   47                   ASRA
7851   47                   ASRA
7852   8B 30                ADDA    #$30
7854   BD BCAB              JSR     $BCAB
7857   C4 0F                ANDB    #$0F
7859   CB 30                ADDB    #$30
785B   1F 98                TFR     B,A
785D   BD BCAB              JSR     $BCAB
7860   39                   RTS
7861                *
7861                CLOCK   FCB     0,0,0,0 clock storage area
7861   00
7862   00
7863   00
7864   00
7865                CHECK   FCB     $50,$60,$60,$13 overflow check
7865   50
7866   60
7867   60
7868   13

CHECK =7865     CLOCK =7861     INIT  =7800     NEWIRQ=7807     NOINC =7820
NXTCHR=782B     NXTDIG=780E     OUT   =784A
```

its normal service routine, thus giving you a permanent clock on screen without interfering with Basic. Figure 4 is a listing of a machine code program which does this.

The locations assigned to ROM, RAM and I/O are fixed by the address decoding of SAM but the use to which RAM and I/O is put is determined by the CPu instructions stored in ROM. The actual functioning of the PIAs has already been discussed in previous editions of *Dragon User* (July 1983 had an article with a few errors which were corrected in March 1984) so the detail has been omitted from the map of Figure 3; however, the usage of RAM gives a good insight into the way Basic actually works and is well worth some investigation.

The area from $00 to $FF is known as page zero and is used by the system to store various system variables. The start and end addresses of any program entered are stored here as well as pointers to the variable storage area and many other pieces of information which collectively decide the current state of Basic. Useful locations are $19 and $1A, which point to the beginning of Basic text; programs can be merged by manipulating these locations as follows:

1) Turn the Dragon off and on to make sure the pointers are set to their default values.
2) Load the first program — this will be stored at $1E00 which is pointed to by $19/1A as a default.
3) Change the pointer as $19/1A to point to the space after the program. The space after the program is the simple variable space pointed to by $1B/1C, so $19/1A should be given this value minus two, to get over the last two bytes of the first program, which are delimiters (see later).
4) Load the second program and renumber it so that the line numbers are higher than those in the first program.
5) Restore the start of program pointer to its original value ($1E00).

The process is accomplished from Basic thus:

CLOAD "first prog" — first program is loaded at $1E00.
POKE25,PEEK(27) — change pointer to end
POKE26,PEEK(28)-2 — of program.
CLOAD "second prog" — second program is loaded after first.
RENUM x,y — change conflicting line numbers.
POKE 25,30 — return pointer to beginning
POKE 26,1 — of first program.

Locations $33 and $34 point to the next DATA statement to read and can be used to provide a kind of random access data statement. Locations $72 and $73 hold the warm start vector; when the reset button on the side of the Dragon is puched location $71 is checked and if it contains $55 and location $72 and $73 point to a NOP (No OPeration — code $12), then control passes to that NOP which normally results in the screen being cleared and the message 'OK' being printed. This vector can be changed to point to your own reset routine; a technique used in Microdeal's *The King* amongst others.

The system also uses the area $100 to $1FF which contains Basic's statement despatch table (more about this later), the board table, the joystick readings and a very useful set of short 'patches' into Basic itself. These patches are three byte subroutines which are called when Basic is executing certain commands. By default these subroutines contain $39 which is the code for RTS (ReTurn from Subroutine), so they don't do anything at all. However, because there are three bytes to each patch, there is room to insert a jump instruction so that when the subroutine is called, it jumps to your own routine which can do something extra before returning. An example of this is intercepting the patch into the LIST command in $1A6 with a delaying routine to slow down listings (see *Dragon User* October 1983 page 51). It is also possible to disable the break key be intercepting the 'read in next statement' patch in $19A and returning with the return address incremented by four, which bypasses the break key check.

The final area of RAM reserved for the system is from $200 to $3FF which is principally used for the cassette buffer.

The next series of locations hold the information which will be displayed on the screen. The area from $400 to $5FF is the default location for the text screen (this can be changed as in the Dragon Toolkit from Premier, which gives 25 different text screens!). The text screen address is put into SAM which tells the VDG which area of memory it should get its data from and what mode it is in. In text mode the data stored in each byte of the screen memory is used to access a character generator which is built into the VDG and which provides the dot patterns which make up each character. The screen code for each charactger is given in the back of the Dragon manual.

After the text screen, the graphics pages start ($600 onwards). The amount of memory used for graphics depends on how many pages have been PCLEARed, the default being four pages, which uses up the area from $600 to $1DFF. IF eight pages are used, the graphics memory extends to $35FF leaving 19k of RAM for your Basic programs and variables, which are stored immediately after the graphics pages up to location $7FFF. If you change the number of graphics pages while you have a program in memory, the system automatically moves the program up or down in memory to follow. In PMODE3, each byte in the screen data represents four dots; the two bits for each dot allow the four possible colours to be encoded. PMODE4 represents each dot on the screen by one bit. As a bit can be either 1 or 0 there are only two colours available in this mode.

After the user RAM area the address space is given over to ROM which stretches from location $8000 to $BFFF. This ROM contains the instructions that enable the Dragon to understand Basic; what follows is a rough outline of how this is accomplished.

When program is typed in, it is stored in the user RAM area according to the start and end of text pointers in zero page. The program is stored as shown:

Storage format of Basic text

1 byte containing 0 in the first program location.

A number of Basic lines stored as:

2 bytes to point to the start location of the next line.
2 bytes for the line number (in hex).
Up to 249 bytes for the line itself.
1 byte containing 0 to mark the end of the line.

And:

2 bytes containing zero to mark the end of the program.

Each character which the machine can display is represented in memory by an eight bit number — it's own ASCII code (ASCII stands for the American Standard Code for Information Interchange). The ASCII code is a standard system adopted by most machines (the Sinclair computers being one notable exception) and is listed in Figure 5. The reserved words, such as PRINT and REM, are not stored in their ASCII formats but are converted to a one byte token, thus cutting down on storage used by a great deal. Functions, such as LEN or ASC, are represented by a token preceded by $FF. Figure 6 contains a list of reserved words and their respective tokens. When the LIST command is used the tokens encountered in each line are looked up in a table and the full word printed out. Converting reserved words to tokens is often known as 'crunching' and detokenising as 'de-crunching'. The routines which do this are at $8F08 and $8F67. As an example of using Basic's routines from your own programs, Figure 6contains a program which produces nearlly formatted listings to a printer using the de-crunch routine from Basic. Looking through the reserved word table turns up a word which isn't explained in the manual. The word is DLOAD and if you enter it into your Dragon, you will get an I/O error. It looks like DLOAD is a relic from Tandy colour Basic, whiere the Dragon Basic has its roots, which has an RS232 (serial I/O) port input command — DLOAD.

When a Basic program is RUN, tokens are used to point into the statement despatch table. This table contains the address of the routine which executes the reserved word designated by the token; looking through the memory map shows where some of these routines are to be found.

The area of RAM between the end of the Basic program and address $7FFF is available for the storage of Basic variables. There are four types of variables in Dragon Basic; simple numeric, simple string, numeric arrays and string arrays and each is stored in a particular format as shown in Figure 7. Simple variables are stored immediately after the Basic program and array variables are stored after these. If a new simple variable is encountered, all the arrays which have been defined are moved up to make room, this process being known as dynamic memory allocation. String variables of both types are stored as pointers to the actual location where the string is stored. If the string is defined in the program text, the pointer will point there. If not, the area at the top end of memory is reserved for strings and the pointer will point to that area. Pointers in zero page are used to keep track of the start and end addresses of the simple variable, array variable and string storage space. See locations $19 to $28 on the memory map.

Figure 5 — The ASCII code

Codes 0-31 are control characters
Codes 128 upwards are not defined (used for graphics in Dragon)

code	char	code	char	code	char	code	char
32		33	!	34	"	35	#
36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+
44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3
52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;
60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C
68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K
76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S
84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[
92	\	93	]	94	^	95	_
96	`	97	a	98	b	99	c
100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k
108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s
116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{
124	|	125	}	126	~	127	

Figure 6 — Tokenised representation of reserved words

Reserved Word	Token	Reserved Word	Token	Reserved Word	Token	Reserved Word	Token
FOR	$80	GO	$81	REM	$82	'	$83
ELSE	$84	IF	$85	DATA	$86	PRINT	$87
ON	$88	INPUT	$89	END	$8A	NEXT	$8B
DIM	$8C	READ	$8D	LET	$8E	RUN	$8F
RESTORE	$90	RETURN	$91	STOP	$92	POKE	$93
CONT	$94	LIST	$95	CLEAR	$96	NEW	$97
DEF	$98	CLOAD	$99	CSAVE	$9A	OPEN	$9B
CLOSE	$9C	LLIST	$9D	SET	$9E	RESET	$9F
CLS	$A0	MOTOR	$A1	SOUND	$A2	AUDIO	$A3
EXEC	$A4	SKIPF	$A5	DEL	$A6	EDIT	$A7
TRON	$A8	TROFF	$A9	LINE	$AA	PCLS	$AB
PSET	$AC	PRESET	$AD	SCREEN	$AE	PCLEAR	$AF
COLOR	$B0	CIRCLE	$B1	PAINT	$B2	GET	$B3
PUT	$B4	DRAW	$B5	PCOPY	$B6	PMODE	$B7
PLAY	$B8	DLOAD	$B9	RENUM	$BA	TAB(	$BB
TO	$BC	SUB	$BD	FN	$BE	THEN	$BF
NOT	$C0	STEP	$C1	OFF	$C2	+	$C3
-	$C4	*	$C5	/	$C6	^	$C7
AND	$C8	OR	$C9	>	$CA	=	$CB
<	$CC	USING	$CD				
SGN	$FF80	INT	$FF81	ABS	$FF82	POS	$FF83
RND	$FF84	SQR	$FF85	LOG	$FF86	EXP	$FF87
SIN	$FF88	COS	$FF89	TAN	$FF8A	ATN	$FF8B
PEEK	$FF8C	LEN	$FF8D	STR$	$FF8E	VAL	$FF8F
ASC	$FF90	CHR$	$FF91	EOF	$FF92	JOYSTK	$FF93
FIX	$FF94	HEX$	$FF95	LEFT$	$FF96	RIGHT$	$FF97
MID$	$FF98	POINT	$FF99	INKEY$	$FF9A	MEM	$FF9B
VARPTR	$FF9C	INSTR	$FF9D	TIMER	$FF9E	PPOINT	$FF9F
STRING$	$FFA0	USR	$FFA1				

Storage format of the four variable types.

Numeric variables

2 bytes containing the ASCII codes for the variable name.

1 byte containing the binary exponent + $81.

4 byte containing the binary mantissa and sign bit.

String variables

2 byte ASCII for variable name — second byte has top bit set.

1 byte containing the number of characters in the string (it is this which limits the maximum string length to 255).

1 zero byte.

2 bytes containing a pointer to the start of the string in memory.

1 zero byte.

The string itself is stored in ASCII at the top of memory.

Numeric arrays

2 bytes containing the ASCII codes for the variable name.

2 bytes containing a pointer to the next array.

1 byte containing the number of dimensions.

2 bytes per dimension containing the number of elements in that dimension.

6 bytes per element of the array. Each element is stored in the same way as a simple numeric variable without the name.

String arrays

2 byte ASCII for variable name — second byte has top bit set.

2 byte pointer to next array.

1 byte containing the number of dimensions.

2 bytes per dimension containing the number of elements.

4 bytes per element of the array. Each element is stored in the same way as a simple string without the name.

There is a more detailed explanation on program and variable storage in the March 1984 issue of *Dragon User*.

Hopefully this article will encourage you to dig a bit deeper into the workings of your Dragon. If so, you might like to acquire some of the following books which I found useful:

Dragon Data's *Information for machine code users* and the Dragon 32 circuit diagrams. These used to be free from Dragon Data — now it is a matter of finding someone who has one.

The *TRS80 Colour Computer Technical Reference Manual* from Tandy. This contains a lot of information relevant to the Dragon.

A book on programming the 6809 microprocessor. I use *The 6809 Companion* by Mike Hames as a quick lookup, and *6809 Microcomputer Programming and Interfacing* by Andrew Staugaard Jr. as a more in-depth text.

Inside The Dragon by Duncan Smeed and Ian Somerville.

Figure 7 — Formatted listing generator

```
787C            100  ****************************************
787C            110  *                                      *
787C            120  *Figure 7 - Formatted listing generator *
787C            130  *                                      *
787C            140  *This program uses BASIC's decrunch    *
787C            150  *routine to produce listings to a printer*
787C            160  *with right-justified line numbers     *
787C            170  *and multi-statement lines split onto  *
787C            180  *separate lines.                       *
787C            190  *                                      *
787C            200  *with prettylist in memory, load the   *
787C            210  *program to be listed and type EXEC&H7800*
787C            220  *                                      *
787C            230  ****************************************
787C            240  *
7800            250         ORG     $7800
7800   86 FE    260         LDA     #$FE      set current output device
7802   97 6F    270         STA     $6F       to printer (-2)
7804            280  *
7804            290  *set up pointers
7804            300  *
7804   9E 19    310         LDX     $19       set X as pointer to BASIC text
7806 10AE 84    320  NXTLIN LDY     ,X        get character
7809 108C 0000  330         CMPY    #0        check for end of program
780D   27 6A    340         BEQ     END
780F            350  *
780F            360  *print right justified line number
780F            370  *
780F   34 20    380         PSHS    Y         save X and Y
7811   34 10    390         PSHS    X
7813   86 20    400         LDA     #32
7815   EE 02    410         LDU     2,X       get line no. in U
7817 1183 2710  420         CMPU    #10000
781B   24 03    430         BHS     NO5
781D   BD 800F  440         JSR     $800F     print a space
7820 1183 03E8  450  NO5    CMPU    #1000
7824   24 03    460         BHS     NO4
7826   BD 800F  470         JSR     $800F     print a space
7829 1183 0064  480  NO4    CMPU    #100
782D   24 03    490         BHS     NO3
782F   BD 800F  500         JSR     $800F     print a space
7832 1183 000A  510  NO3    CMPU    #10
7836   24 03    520         BHS     NO2
7838   BD 800F  530         JSR     $800F     print a space
783B   12       540  NO2    NOP
783C   EC 02    550         LDD     2,X
783E   BD 957A  560         JSR     $957A     print line number
7841   86 20    570         LDA     #$20
7843   BD 800F  580         JSR     $800F     print a space
7846   35 10    590         PULS    X
7848            600  *
7848            610  *print the line
7848            620  *
7848   BD 8F08  630         JSR     $8F08     decrunch BASIC line
784B   8E 02DD  640         LDX     #$02DD    point to decrunched text
784E   A6 80    650  NXTCHR LDA     ,X+
7850   4D       660         TSTA              check for end of line
7851   27 1D    670         BEQ     EOL2
7853   81 3A    680         CMPA    #$3A      check for a colon
7855   26 14    690         BNE     OUT
7857            700  *
7857            710  *
7857            720  *
7857            730  *
7857            740  *
7857            750  *print new line for multiple statements
7857            760  *
7857   86 0D    770         LDA     #$0D      print (return)
7859   BD 800F  780         JSR     $800F
785C 108E 0005  790         LDY     #5        print 5 spaces
7860   86 20    800         LDA     #32
7862   BD 800F  810  NXTSPC JSR     $800F
7865   31 3F    820         LEAY    -1,Y
7867   26 F9    830         BNE     NXTSPC
7869            840  *
7869            850  *
7869            860  *
7869   86 3A    870         LDA     #$3A      print a colon
786B   BD 800F  880  OUT    JSR     $800F
786E   20 DE    890         BRA     NXTCHR
7870            900  *
7870            910  *end of line (return)
7870   86 0D    920  EOL2   LDA     #$0D      print (return)
7872   BD 800F  930         JSR     $800F
7875   35 10    940         PULS    X         restore pointer
7877   20 8D    950         BRA     NXTLIN    go for another line
7879            960  *
7879   0F 6F    970  END    CLR     $6F       restore current device
787B   39       980         RTS

END    =7879    EOL2   =7870    NO2    =783B    NO3    =7832    NO4    =7829
NO5    =7820    NXTCHR=784E     NXTLIN=7806     NXTSPC=7862     OUT    =786B
```

Memory Browser

THIS short program allows one to browse through the memory of the Dragon. After entering the start address, which can be in either in hex or denary, the contents of 60 bytes from that address are displayed, together with the equivalent ASCII character. Any non-ASCII or control code characters are displayed as full stops. The user can then browse backwards or forwards through the memory by using the up and down arrows, or restart from a new address by touching the spacebar. Interesting sample browses start at hex 1E00 (Basic program area, unless you have PCLEAR), and hex 8000 (the BASIC keywords).

The superiority of Dragon Microsoft BASIC was simply demonstrated when I tried to adapt this program for the Commodore 64: no PRINT USING, no HEX$(n), no PRINT£ etc made it half as long again!

In retrospect, it's a pity that Dragon Data didn't make better play of the commonality of Dragon Basic with that supplied to IBM for the PC and XT, both highly respected machines.

Hugh Petfield
Room 216
Blandford House
Melbury Terrace
London NW1 6LB

```
100 A$=CHR$(128):S$=STRING$(32,A$):REM Memory Display by H Petfield
110 B$=A$+"addr"+A$+"00"+A$+"01"+A$+"02"+A$+"03"+A$+"04"+A$+"05"+A$
+A$+"012345"+A$:REM TYPE addr IN REVERSE VIDEO
120 C$=A$+"%  %"+A$+"%%"+A$+"%%"+A$+"%%"+A$+"%%"+A$+"%%"+A$+"%%"+A$
+A$+"%    %"+A$
130 D$=A$+"hex"+STRING$(22,A$)+"ascii":REM TYPE hex AND ascii IN RE
VERSE VIDEO
140 CLS:INPUT "START ADDRESS>";A
150 CLS0:PRINT B$;D$;S$;
160 PRINT@452," USE UP & DOWN ARROWS,";
170 PRINT@484,"OR SPACEBAR TO RESTART";
180 PRINT@96,"";
190 FOR ROW=0 TO 9
200   E$=HEX$(A+(ROW*6))
210     IF LEN(E$)<4 THEN E$="0"+E$:GOTO 210
220   F$="......"
230   FOR COL=0 TO 5
240     F(COL)=PEEK(A+(ROW*6)+COL)
250       IF (F(COL)<32 OR F(COL)>127) THEN 270
260     MID$(F$,COL+1,1)=CHR$(F(COL))
270     G$(COL)=HEX$(F(COL))
280       IF LEN(G$(COL))=1 THEN G$(COL)="0"+G$(COL)
290   NEXT COL
300   PRINT #-Z,USING C$;E$,G$(0),G$(1),G$(2),G$(3),G$(4),G$(5),F$;
310 NEXT ROW
320 P$=INKEY$:IF P$="" THEN 320
330   IF ASC(P$)=94 THEN A=A-48:GOTO 180
340   IF ASC(P$)=10 THEN A=A+48:GOTO 180
350   IF ASC(P$)=32 THEN 140
360 CLS:STOP
```

Intelligent Pattern Generator

THIS has been written in response to all those letters asking for short programs. It will fill the screen with a random pattern, and is best viewed on a colour TV or monitor.

Line 1 and 2 set the random co-ordinates; lines 3 and 4 work out where to put the patterns; line 5 and 6 draw the patterns to the screen; lines 7 and 8 move through the various screens.

R. Bailey (G4PPP)
52 Princess St.
Chase Terrace
Staffs WS7 8JN

```
1 PMODE4:SCREEN1,0:PCLS:C7=RND(-TIMER)*2.5+.1
2 A=RND(50)+8:B=RND(40)+10:D=RND(40)+10
3 W=255-(D*2):W1=INT(W/A):W2=W1*A:W3=(W-W2)/2
4 Q=191-(D*2):Q1=INT(Q/B):Q2=Q1*B:Q3=(Q-Q2)/2
5 FORX=W3+D TO 255-W3-D STEP A:FORY=Q3+D TO 191-Q3-D STEP B
6 CIRCLE(X,Y),D,,C7:NEXT Y,X:FOR H=3 TO 4
7 FOR J=0 TO 1:PMODE H:SCREEN 1,J:FOR Z=1 TO 999
8 NEXT Z,J,H:SOUND 100,3:GOTO 1
```

Expert's Arcade Arena

Write to 'The Expert' at Dragon User
12-13 Little Newport St, London WC2H 7PP.
with all your arcade tips and hints.

AND so the new year dawns, bringing with it new hope (unless you own a Dragon, in which case you've probably completely given up hope!), new life, new civilisations, boldly going where no man has, erm sorry I appear to have lost track here a bit.

Anyway, the new year heralds something else, the new year brings another TWELVE MONTHS of the award winning EXPERT'S ARCADE ARENA. Yes, we've won again, last night at the prestigious AFCCEAA awards ("Awards For Columns Called Expert's Arcade Arena") which took place in my bedroom, this column took every award available. Yes, you are reading the "Best Monthly Column In Dragon User Called Expert's Arcade Arena", the "Most Witty Column Called Expert's Arcade Arena", and last but by no means least the "Most Arrogant And Clever Clever Column With The Most Awards Ever In A Magazine Called Dragon User".

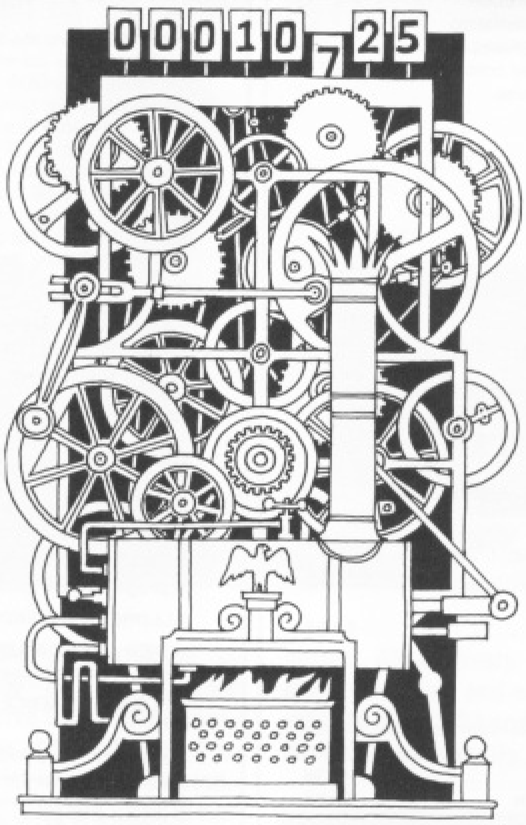
So, a new year, but all the same old jokes. As I'm writing this column in November and it's not really the new year yet, I haven't received your Christmas gifts yet,but a big thank you in advance for them and the winner of the "Really Rather Silly Present" competition will be announced soon!!

Now then, if you remember a few months back I published a complete list of the messages at the end of each screen on Time Bandit and asked you to send me a complete list. Now, I wasn't really that convinced that anyone would be stupid enough to copy out a list and post it in to me (*Don't be so sure. I copy out nearly everything you write and post it out to the entire readership — Ed.*) but I thought it was worth a try for a good laugh and YES!! Someone actually bothered to do a competition with no question! SO, I am proud to announce that Philip Saffer receives the award for "Possibly the most stupid person in the world who owns a Dragon and reads Expert's Arcade Arena and has nothing better to do than copy out lines of text from a column called Expert's Arcade Arena in a magazine called Dragon User" (*Yes, this sounds very familiar — Ed.*).

Philip has also sent in some pokes to help those among you who aren't smart enough to play games by yourselves ...

DRAGON HAWK: "Load the game with the following command: CLOADM "", 32000 : POKE 32184,57: EXEC 32159 then when the game has loaded type POKE 11590, 127 : EXEC 4096
.... unfortunately all of the rest of Philip's pokes were covered by me in my last column, so tough titty there Philip! Happy Christmas!

OIKMM. Now then, on the subject of last month's issue, if you'll all take it out and look on page 4 in the bottom left hand corner you'll see an apology, now then, the last line reads "Dragon User and Mr. Barclay would like to apologise to all concerned for this misunderstanding." You notice it DOESN'T



say "Dragon User and Mr. Barclay and The Expert would like to apologise to all concerned for this misunderstanding". Well, in case you were wondering, I'll tell you: it's because I'm NOT SORRY AT ALL! In fact, I find it RATHER FUNNY! However, I would like to apologise on behalf of my editor (and I say "my" because I do in fact own her)for the fact that a map of the game Fantasy Fight was called Fantasy Flight, the Map! (*Please refer above for explanation. And he means 'owe' — Ed.*). I can assure my loyal readership that Ms. Armstrong has received 40 lashes as well as being tied .... sorry, got off the track again there.

Some of you may be wondering "Where is the map this month Oh great one??" Well, the truth is, there ISN'T ONE, so you can stop looking for it. Instead, next month's Arcade Arena will be a MAP SPECIAL with maps of *Brock's Kingdom*, stages two and three of *Copta Snatch* (with four and five to follow in March), *Tanglewood* and a rather incomprehensible map of Universe One to *Total Eclipse*. I warn you in advance that this one is going to be difficult to read, but I've decided to publish it because it's so darn useful!!

Now then, to Joe Brincat from Malta who really sent probably the best written letter I've ever seen ... (which reminds me, I must get onto the new competition) and he asks why INVEST and SALVAGE on *Total Eclipse* are always out to lunch. The answer is simply because those options are not available until later Universes.

Now then, to the competition ... we are still running the following competitions most of which are free to enter and have sumptuously non-existant prizes:

What is the best music to play to? (Latest nomination is Keith Jarrett, by me!)

What are the codes at the end of each screen on *Time Bandit*? (well, it's got to be worth a try, hasn't it?)

Who can send the silliest Christmas present to the Expert?

Who is the Expert, and who does he or she have such strange musical tastes?

AND NOW ... THE NEW COMPETITION...

Who can send the best written letter to the Expert? (I mean in handwriting, not use of English!)

Right, now to moved on to a letter from M. "No, I'm Not A Software Pirate Really Your Honour" Vine, who asks for more maps (your plea is answered) and provides more pokes (hmm, strange contradiction somewhere there) so that those of you who don't want to play in black and white can move up to glorious full colour. The addresses are:

```
AQUANAUT 471: POKE 15309,224
TREKBOER: POKE 7310,224 (or 232)
VORTEX FACTOR: POKE 7963, 224
SHENANIGANS: POKE 7263, 224
ZAXXON (from SEGA): POKE
25230, 224 (or 232)
```

To use these pokes you need Paul D. Burgin's loader programs which have been the cause of more phone calls and letters than any other, and for the first time (and only time until January 1988) here are the infamous programs A, B and C:

PROGRAM A:

```
10 A=100: POKE A,142: POKE A+1,4:
POKE A+2,0: POKE A+3, 126: POKE
A+4, 183: POKE A+5, 91: SKIPF:
EXEC A
```

PROGRAM B:

```
10 A=30000: POKE A,189:, POKE A+
1, 160: POKE A+2, 146: POKE A+3,
134: POKE A+4, 57: POKE A+5, 183:
POKE A+6, 1: POKE A+7, 103: POKE
A+8, 57: EXEC A
```

PROGRAM C:

```
10 PCLEAR1: CLEAR 10: FOR I=1800
TO 1851: READ M: POKE I,M: NEXT:
EXEC 1800 20 EXEC PEEK
1546)*256 + PEEK (1547)
30 DATA 134, 126, 183, 1, 142, 48, 140,
28, 191, 1, 143, 141, 31, 141, 34, 134, 3,
151, 0, 142, 64, 0, 141, 9, 10, 0, 38, 247,
190, 64, 12, 159, 0, 126, 183, 91, 158, 0,
141, 7, 141, 2, 53, 144, 142, 6, 0, 159,
126, 126, 185, 51
```

Right, that's it for this month, keep the letters coming, see you soon, etc...etc...etc...

Functions

I HAVE had problems converting ATN functions to ACS and ASN (Arc Cosine and Arc Sine). Those in the manual appear to be incorrect; please could you give the correct versions.

J. F. Greenwood
91 Keyhaven Road
Milford On Sea

YOU can use the following function definitions to return the arc sine and arc coosine of x respectively:
DEF FNS(X)=ATN(X/SQR(-X*X+1))
DEF FNC(X)=-ATN(X/SQR(-X*X+1))+1.57



Restore

I AM at present writing a game for my Dragon 32. However, I have come across a problem. Is it possible to extract separate pieces of information from a DATA statement at random? If there was a RESTORE N command it would help tremendously.

Stuart May
67 Rydens Road
Walton on Thames

ALTHOUGH Dragon Basic does include the RESTORE command there is no facility to restore to an individual line. I gave a short routine to do this in a recent DU. Alternatively, read all your data into an array at the start of the game, and access it directly in the array. This will generally be faster than using READ/DATA. A hint if you must repeatedly use READ/DATA: put all the DATA statements at the start of the program.

Not ready

I AM writing a wordprocessor program for my Dragon 64 with DragonDos, but I have run into a problem. How to access the 'Ready' and 'Write Protect' lines of the drive to see if there is a disc present in the drive.

The DragonDos routines just lock up if no disc is present, so how can I prevent this?

Sue Edwards
Solihull
West Midlands

UNFORTUNATELY, DragonDos does not support the 'ready' line from the disc drive, and so this cannot be used to check if the drive is ready. However, it is possible to detect if a disc is in the drive and ready by inspecting the Index strobe input.

Initially the DOS routine at 49509 should be called, this returns the drive status in the command register at $FF40. If bit 6 is set then the disc is write protected. If bit 2 is zero then the drive is not on line and is hence not ready. Bit 1 is the index input, this goes high when the index hole passes under the led in the drive. Your machine code routine should look at this in a loop (about 1000 times say) and see if it changes — if it does then a disc is in the drive and the hatch is closed. This should be done with interrupts disabled.

Link up

I OWN a Dragon 64 and two printers, one with serial interface. I have been trying for sometime to link up the serial printer to the Dragons RS232 port so that I can run both printers at once.

I have tried connecting the following lines, GND to GND DATA to RX, and STATUS to CTS. The problem is that the Dragon just freezes up whenever I send a character to the printer. Examining the value at &HFF05 seems to indicate that the printer is permanently busy?

Ian Thomas
Solihull
West Midlands

THIS IS quite a common problem on which I regularly get letters. The problem is that the output 'status' from the serial pot on many printers goes 'high' (+12v) when the printer is busy and 'low' when ready. The Dragon's 'CTS' expects the opposite to this, ie, high when ready, low when busy.

Unfortunately, there is no simple solution to this. You could build a small interfact consisting of an hex invertor chip to invert the status output. Alternatively, you can try increasing the end-of-line delay value at 1021/1022 and tieing 'CTS' to +12v permanently.

Printer

THE PROBLEM is that I have decided to get a printer for my Dragon 32. However, as you can understand, the problem of which one to choose is extremely difficult.

One machine has come to may attention above all others, this being the DMP-2000 from Amstrad. it is within the price range I'm looking at and has a near letter quality print which I need.

With this in mid my question is quite simple. Will my computer be able to use this printer, or are the two machines incompatible.

George Cooke
54 Sleetmoor Lane
Somercoates
Derbyshire

THE AMSTRAD DMP-2000 has a standard 8-bit Centronics parallel interface and will work quite happily with a Dragon computer. IN fact, because of the Dragon's 8-bit interface, it is possible to directly access the extended characters available on the printer, which cannot be done from an Amstrad micro (which have 7-bit interfaces).

You will need a suitable printer lead, any Dragon lead should work, but the one supplied with the printer will only fit Amstrads.

Pins

AT SCHOOL I am about to do a Computer Studies Project. I would like to do my project on my Dragon 32 as it is much better than the school computers, but I do not have a printer. My teacher could put together a makeshift printer lead to use with a school printer, but he needs to know the separate pin connections for the Dragon's printer port.

Could you give me this information, or tell me where I can obtain it from?

R. Northcott
1 Elmuil Road
Tewkesbury

THE pin connections for the printer port can be found at the back of most Dragon manuals, or in the separate 'Additional Information' booklet. However, this question does come up quite regularly, so I'll repeat it here.

Odd numbered pins on top line, pin 1 on the right (viewed end on), even pins on bottom, pin 2 on the right again.

PIN 1 Print Strobe PIN 2 +5 volts
3 Data bit 0 4 +5 volts
5 Data bit 1 6 Ground
7 Data bit 2 8 Ground
9 Data bit 3 10 Ground
11 Data bit 4 12 Ground
13 Data bit 5 14 Ground
15 Data bit 6 16 Ground
17 Data bit 7 18 Ground
19 ACK (do n? use) 20 BUSY

Database

I AM writing a database program in Basic and use a Dragon with twin disc drive. I want to 'Fread' and 'Fwrite' to the drive not being used by the menu program (which will RUN "DATABASE" etc.).

However I find that the command "DRIVE 2" when included in a program causes an ?SN ERROR. Is there a way of changing the drives without having to stop the program?

Arthur Wrennall
20 Railway Road
Darwen

THE DRIVE command shouldn't cause you any problems when used as described above. However, it will not accept a variable as a paramenter, eg DRIVE D. The drive may be included as part of the FREAD/FWRITE filename, so a variable could be used thus:
FWRITE CHR$(48+D)+":FILENAME.DAT" ;variable list

Alternatively, you can set the default disc drive to be used with the following direct poke to DOS RAM:
POKE 235,D (Where D is 1-4)

Winners and Losers

Every month, Gordon Lee will look at some prize programming points from a previous month's competition

OK, you lucky lot! Here is your opportunity to share in some feedback from Dragon User's regular competition pages. For an experimental period, the Powers That Be *(We call them the Powers That Do around here — Ed.)* have asked me to sift through the entries to the competition answers and highlight some of the innovations in programming and problem solving that competitors use. These should provide a useful reference for use in programming generally, as well as for tackling future competitions (I can see that I shall have to be even more devious...). In addition, mention of some of the commonly occurring mistakes should be of help in avoiding these pitfalls, expecially for beginners who may be encouraged to 'have a go' at the competitions, and maybe win some prizes.

This month, we'll be looking at the August '86 competition in greater detail, and, as with most of these problems, there are four main stages in their solution:

i) **Read the question carefully**. How many times have we heard that bit of advice? For the August problem you had to substitute digits for the letters in DRAGON/USER, such that the result was a two-digit number, the cube of which spelled a common English word when the digits were replaced by letters from the original sum.

A surprisingly large number of competitors were of the impression that you were not permitted to use a zero as one of the digits. There was nothing in the question to suggest this, although in the event, this would not have affected the result which, purely by chance, did not contain a zero. Had it done so, many of the programs submitted would not have come up with the answer. The question of a 'leading' zero is a different matter. Conventionally, in puzzles of this type, it is assumed that a leading zero is not permitted, unless specifically mentioned in the question. Thus, in this particular question, neither the 'D' nor the 'U' can be replaced by a zero, since it would seem reasonable to assume that, for example, 'DRAGON' was a six-digit number and not a five-digit number with a zero in front.

A number of entrants allowed values with a leading zero which, apart from increasing the number of tests to be carried out, could have produced spurious results.

ii) **Method of approach**, or the interpretation of the problem into a program which the computer can handle. Just as there is more than one way to crack a nut so there may well be alternative ways, often as good, maybe better, to solve a given problem. Nearly all competitors elected, probably wisely, to take as a starting point the possible values of 'USER' and the two-digit quotient, and then work backwards to find and test the corresponding value for 'DRAGON'. Most entrants using this approach generated the value for 'USER' in a single FOR/NEXT loop, testing each value for duplication of digits, but some used four FOR/NEXT loops to generate and test each digit individually before combining them into the final four-digit number. Which of the two methods will be most time-efficient is best determined by experiment. The long running time of the programs was a generally expressed comment by many readers, so any opportunity to reduce this should be seized. However, it is probably more efficient to exclude whole ranges of impossible values for testing rather than to test every set of values for comparatively minor features. The listing on this page is a 'composite', compiled from a number of listings submitted, but which follows the general ideas of the majority of the programs examined. It runs for several hours, but the winning answer appears after about three! One entrant to an earlier competition set in *Dragon User* admitted to a running period of over a week! Clearly, it's a case of practicalities and, being of a lazy nature myself, I see no reason why a computer should not do most of the work — that's what they are there for, isn't it?

Most entrants took the minimum and maximum values of 'USER' to be 1024 and 9876 respectively. In fact, 'USER' must be greater than 2013 as any smaller value, multiplied by an two-digit number will give as a value for 'DRAGON' either a five-digit number or, at best, a six-digit number beginning with a '1'. Clearly, if the 1 already stands for the 'U', then it cannot also represent the 'D'. THe range to be included can be further reduced in many cases. If 'F' is the two-digit result of DRAGON/USER, then for any value of 'F', the range of 'USER' will lie between INT(102345/F) and INT(987654/F). Other points which can be incorporated into the program are that F cannot be a multiple of ten or its cube will end in three zeros, and hence the resulting word in three identical letters. Also, 'R' cannot equal zero, otherwise 'N' would also have to equal zero if the quotient is to be a whole number. From this analysis of the problem we should now be in a position to:

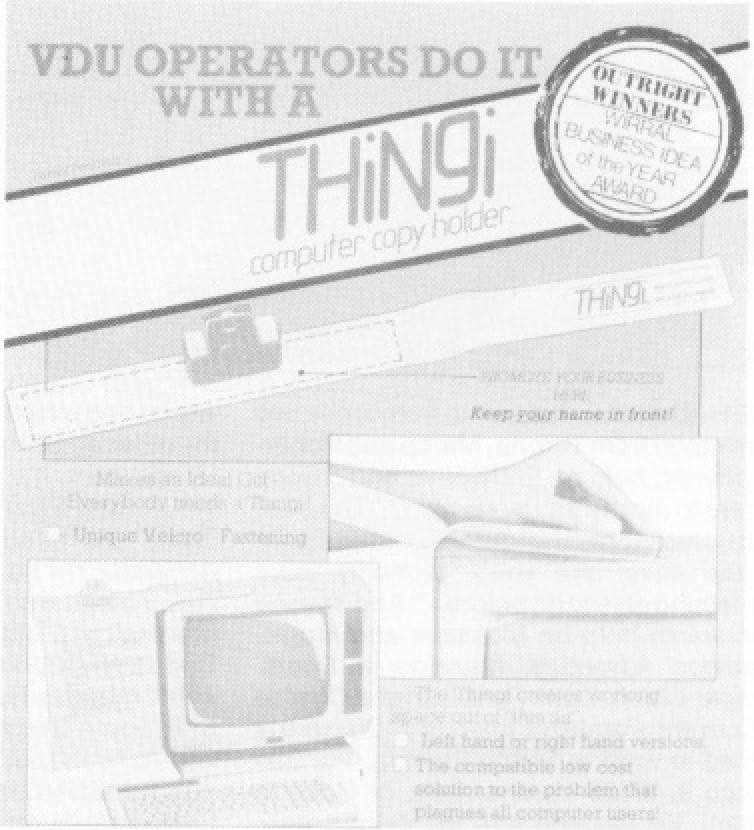
iii) **Write the program**. The listing given here will print out all sets of values in which the cube so found has all its digits present in the generating division, the letter equivalents being printed alongside. A few (very few) readers also tried to quicken things along by using the notorious Dragon speed poke, &H FFD9,0. The effectiveness of the individual programming submitted was reflected in the number of possible 'words' that were printed out for examination. Where included, these varied from a list of under a dozen words to several with over seventy. Also the length of the programs ranged from under 14 to over 80 program lines. Once these lists have been printed out, the final (and comparatively easy) stage is to:

iv) **Check the results**. In the case of the competition in question this is simply to find a common English word from amongst those letters sets printed out. E. A. Newman of Surrey reports that the letter set ORNNN occurs twice (has this word any special or mystical significance, E.A?), while C. Hitchinson of Middlesbrough also lists a number of alternative solutions. These include such words as aarnra, eauoeo, ooades and rerrr, which he describes as 'words used in comics to give sound effects to heads being crushed, people being tortured or mercilessly beaten, and aircraft being shot down'. Strange reading matter you have there!! Phil Sapiro of Liverpool was surprised to find that there was only one acceptable word formed from the cubes tested. In setting the question, I had first tested the squares without finding a single word. As this type of letter substitution is almost the equivalent of a pseudo-random selection of letters from the words 'DRAGON USER' the chance of an English word occurring at all must be fairly low. Perhaps interested readers would like to generate some random sequences and find the incidence of acceptable words.

Finally, Martin Vango of Banbury deserves mention for the month's worst double pun used in the tie-breaker sentence.

'You're never alone in the Universe... Because Mars is always got Orion you!' Go on! Say it quickly! Now I know what made Banbury cross!

```
10 J$="DRAGONUSE"
20 FOR F=11 TO 99
30 IF F/10=INT(F/10) THEN F=F+1
40 MX=INT(987654/F):IF MX>9876 THEN MX=9876
50 MN=INT(102345/F):IF MN<2013 THEN MN=2013
60 FOR USER=MN TO MX
70 IF USER/10=INT(USER/10) THEN 220
80 U$=STR$(USER):U$=MID$(U$,2)
90 Z$=U$:GOSUB 250:IF FL=1 THEN 220
100 DRAGON=USER*F
110 D$=STR$(DRAGON):D$=MID$(D$,2)
120 IF MID$(D$,2,1)<>MID$(U$,4,1) THEN 220
130 Z$=D$+LEFT$(U$,3):GOSUB 250
140 IF FL=1 THEN 220
150 S=F*F*F:S$=STR$(S):S$=MID$(S$,2):T$=D$+U$:W$=""
160 FOR V=1 TO LEN(S$)
170 FG=0:FOR W=1 TO 9
180 IF MID$(S$,V,1)=MID$(T$,W,1) THEN W$=W$+MID$(J$,W,1):FG=1
190 NEXT:IF FG=0 THEN V=LEN(S$)
200 NEXT:IF FG=0 THEN 220
210 PRINT D$;" ";U$;" ";W$
220 NEXT USER
230 NEXT F
240 END
250 L=LEN(Z$):FL=0
260 FOR G=1 TO L-1
270 FOR H=G+1 TO L
280 IF MID$(Z$,G,1)=MID$(Z$,H,1)THEN G=L-1:H=L:FL=1
290 NEXT:NEXT
300 RETURN
```



VDU OPERATORS DO IT WITH A
THiNGi
computer copy holder
OUTRIGHT WINNERS
WIRRAL BUSINESS IDEA of the YEAR AWARD
THiNGi
Keep your name in front!
Unique Velcro Fastening
Left hand or right hand versions
The compatible low cost solution to the problem that plagues all computer users!





I'VE HAD a letter this month from those two gentlemen from Verona, Marco and Roberto, offering help on *El Diablero* and others. Where to dip the infamous twig? "The oil of cactus blossoms is under the golden statue in the cave, but you'll probably have something else to do before..." And anyone who wants help on *Monsters and Magic, Poseidon Adventure, Mansion of Doom, Danger Island, Don't Panic, Crystal Chalice of Quorum, Franklin's Tomb* or *El Diablero* is invited to write to Marco Faedda, Via San Rocco 9, 37125 Verona, Italy, or Roberto Vallani, Via Favretto 9, 37100 Verona, Italy. Maybe they can also give you some tips on where to spend your summer holiday this year.

Frequent updates

Someone who sounds like he needs a holiday is Mr D.King of Basingstoke, who signs himself "A nagged father". He's nagged because his children have just started playing adventures and he's expected to know all the answers. Now that he's got copies of my clue sheets he's probably a bit better informed, and I think it's about time I informed everyone again as to what's available, as I frequently get letters from people asking for regular updates on this, and I've also just prepared a solution sheet to *Vortex Factor*, thanks to Simon Hargrave, who's becoming such a regular in this column that I'm thinking of charging him rent.

Anyway, in addition to *Vortex Factor*, for the price of a stamped addressed envelope you can have copies of any or all of the following help sheets/solutions: *Black Sanctum, Cricklewood Incident, El Diablero, Fishy Business, Franklin's Tomb, Juxtaposition, Madness and the Minotaur* and *Trekboer*. In addition, I've a copy of the notes that were on the original cassette inlay for *El Diablero*, and for those who don't want to leap straight into full solutions there's a sheet with a list of the various verbs recognised by *El Diablero* and *Trekboer*.

Chris Morris of Bath could do with a few of those, I think, but lack of SAE means I'll deal with his questions here, as per usual. In *Trekboer* to get past the forcefield you need TELUMA EHT and you find this on top of HPATONEC EHT, while to get over the lava you ETARG HGUORHT OG and then ELBBUR SSORC. For your problem on *Vortex Factor* that won't budge, the answer is that it won't budge. To light the lamp in *Golden Baton*, the only problem is that the matches are on one side of the moat and the lamp on the other, so SEHCTAM WORHT and TAOM MIWS, then it's a straightforward PMAL THGIL and PMAL THGILNU. Outside the castle you need to EPOR WORHT. For *Aquanaut 471* you'll need to write to some clever clogs who's solved it, such as Keith Marshall of 5 Maidstone Road, Lowestoft, Suffolk NR32 2AY or Stephen McMahon of 146 Kimberley Road, Penylan, Cardiff CF2 5AF, South Glamorgan.

Stephen's just written to me to say that he's solved *Aquanaut 471*, and as well as offering help he would therefore like to swop it for an adventure he doesn't have, such as *Vortex Factor, Cimmeon Moonor* other graphics adventures apart from *Trekboer, Juxtaposition, Syzygy, Ring of Darkness* or *Total Eclipse*. He also needs help on destroying Darth Vader. Any offers?

Also in swop mood is Ann Cooper, Site House, Ingoldmells Holiday Park, Sea Lane, Ingoldmells, Lincs PE25 1PG. I don't think I'd normally encourage people to go rampantly swopping software instead of buying it, but as Dragon adventures are getting harder to come by all the time, particularly the older titles, I think a bit of trading won't go amiss, as long as it's legitimate swopping of originals and not copies. With adventures it often is a case of not wanting to play it again once you've cracked it. If you send SAE to Ann she'll list what she's got, which she says includes several quite old titles plus a few lesser-known American ones. She also offers help on *Sea Quest* and *Shenanigans*, which gives you some idea of two she might be ready to part with.

Robert Margrave is ready to part with copies of his adventure, *Fernando Poo*, having decided to market it himself at a price of £3, or £2 if you provide a blank tape and SAE. At either price it's a bargain, and I can only repeat what I've said in earlier columns which is that Rob's adventure is one of the best I've ever seen sent in by a reader, and it would surely have been published in days when the Dragon was a healthier beast. Rob's address is 24 Canon Young Road, Whitnash, Leamington Spa CV31 2QU.

Phil Callaghan of Stafford sends in some hints on *Trekboer*. Can't cross bridge? TIUS ECAPS EVOMER. Can't find Xendos? NOTTUB DER EHT HTIW ROOD EHT HGUORHT. Can't get out of chasm after getting out of grate? EERT OT EPOR EIT.

No clues from Carl Truett, just the information that he's finished and is prepared to help other readers on: *The Cricklewood Incident, Vortex Factor* and *Aquanaut 471*. "*Vortex* is a brillant adventure," Carl reckons, "I would recommend it to anyone, but I must agree with you about *Aquanaut 471* and the irritating 'arcade' screens. Nice graphics, but little else." Carl's address for help is 11 Hayden Road, Rushden, Northants NN10 0HX.

Simon Scoltock signs himself "A reader without shame," and the reason for this is that he gives various ways of cheating in Richard Shepherd's game of *Transylvanian Tower*. Knowing how many of you readers out there are also without shame, and believing that all is fair in love and vampire hunting, here is what Simon says. If you're stuck on a particular level then BREAK into the program and type LV=? (ENTER) CONT (ENTER). Instead of the ? type the number of the level you want to resume playing on. The same method works if you want to change the number of bats that you've killed, in which case type BA=? and CONTinue, while you can increase your number of bullets by altering the BL variable. To get a longer look at the map press SHIFT + @ at the same time to pause and then any key to continue. And what do you need to kill Count Kreepie? The answer's the name of a daily paper, and it isn't the Transylvanian Times.

Lost in space

I'd like to thank Bob Wilkinson for sending me four photocopied maps of *Juxtaposition* for passing on to other readers, and there are now four readers somewhere who are grateful to Bob for his kindness. Others will be grateful if they're stuck on that game, *Trekboer, Franklin's Tomb, Mansion Adventure* or *Black Sanctum* (text version), as Bob's offering help on those, as well as adventures he's partially completed, like *Lost in Space, Fishy Business, Pettigrew's Diary, Cricklewood Incident* and *Syzygy*. As for one reader's comment that the Adventure Trail heading should be removed to make the column bigger, Bob says "Hands off! The *Adventure Trail* heading is the first thing I look for when I open Dragon User." This man of taste lives at 16 Pound Street, Warminster, Wilts BA12 8NL.

Another kind creature is H.J.Nieuwenburg, 3 Long Drove, Amberhill, Boston, Lincs PE20 3RQ. This reader's sent in the

security code for the Phoenix Software game, *The Emperor Must Die*, which had what I always thought was a rather silly idea of making you complete an arcade game before you got the pass-code to the adventure in the twin-game pack. If the arcade game's too easy it's a waste of time, and if it's too hard then adventure fans will never get through it. Anyone in that position, please note that the code is (printed forwards this time): TCHK. The same reader offers help on *Vortex Factor*, and would like to hear from anyone who's ever come across any adventures in Dutch.

Andrew Webb of Wells admits that most adventures are double-Dutch to him, and he reckons if there was a Eurovision contest for the most failed adventurer, he'd win by a mile. He's presently stuck in six of them (that's nothing, Andrew, believe me!), and I've tried to de-stick him in my reply, though he can offer help on the three adventures he's actually solved, *Ring of Darkness*, *Mansion Adventure* and *Calixto Island*. Andrew's partly made progress on *Syzygy* by cheating — does this column have no honest readers at all?? He's discovered that if you pause the cassette two or three times while loading in a saved game, you can sometimes find yourself in a much better position than you were previously, though this doesn't always work. Andrew's address is 4 St Cuthbert Way, Wells, Somerset BA5 2JL.

Keith Emmerson wants to see his name mentioned in *Dragon User*, so obviously the rest of his life is now going to be something of an anti-climax after his moment of glory. th seems to have an unhealthy interest in obscure adventures (not to mention obscure football teams like Stockport County) and wants to hear from anyone who can help him on *River of Fire* and *To Boldly Go*, and also wonders where he can get hold of copies of two titles from ASP, *Conquering Everest* and *The Valley*. In addition, he can offer help on *Superspy, Movie Producer, Franklin's Tomb, Island Adventure, Dragon Mountain, Poseidon Adventure, Syzygy, Castle Adventure* and a typed-in listing called *Adventure* from some little-known rag called *Your Computer*. The Emmerson abode is at 31 Waverton Avenue, Heaton Chapel, Stockport, Cheshire SK4 5JT.

From rather further afield than even exotic Stockport comes Guido Jongbeis, Van Loenoutstraat 50, 2100 Deurne (Antwerp), Belgium. Guido's also got some pretty obscure adventures, and can answer questions or provide complete solutions on *Mansion of Doom, Jerusalem Adventure, Rescue on Alpha II, The Head of the Beast, Ring of Darkness, Trekboer, Syzygy* and *The Curse of Abu Simbel*. He's also sent in lots of tips, and I'll be raiding those in future issues — time for a few more clues soon, I feel.

Finally from Richard Heath in Leek, various questions, including what use is the girl in the *Juxtaposition* discotheque? The answer to that is: RENTRAP GNICNAD DOOG A SEKAM EHS. And that's all. In more ways than one!

Adventure Contact

To help puzzled adventurers further, we are instituting an Adventure Helpline — simply fill in the coupon below, stating the name of the adventure, your problem and your name and address, and send it to Dragon User Adventure Helpline, 12/13 Little Newport Street, London WC2H 7PP. As soon as enough entries have arrived, we will start printing them in the magazine.

Don't worry — you'll still have Adventure Trial to write to as well!

Adventure ....

Problem ....

....

Name ....

Address ....

....

Adventure Contact

Adventure: Caverns of Chaos
Problem: How to get past the Terrible Toy Factory screen
Name: Mabon John
Address: Y Bryn, Bull Lane, Denbigh, Clwyd, N. Wales LL16 35N

Adventure: Jerusalem/Ultimate
Problem: Can't do anything, need maps.
Name: Tudor Davies
Address: 57 Sunnyside Road, Weston Super Mare, Avon BS23 3QD

Adventure: Juxtaposition
Problem: Every time I try to go in to Baron White's fort, I get the reply "Not now." Why?
Name: Martin Lewis
Address: 30 London Road, Cirencester, Glos. GL7 1AE

Adventure: 1) Mountains of Ket 2) Temple of Vran
Problem: 1) Where are the hat and the cloak? 2) How can I open the huge doors?
Name: Dario Palmieri
Address: Via Molino Di Peslarola 12, 40131 Bologna, Italy

Adventure: The Vortex Factor
Problem: How can I open the safe? How can I open the door?
Name: Josep Jame
Address: Cl Jaun Moragall No. 5, 2o2o, 08800 — Vilanova, la Geltni (Barcelona).

Adventure: Juxtaposition
Problem: What use is the girl in the Lasertheque. How do I get the spade from the Droid in the Red Mines
Name: Tim Entwhistle
Address: 8 Snapps Close, Wroughton, Swindon, Wilts SN4 0RN.

HERE'S MY CLASSIFIED AD.
(Please write your copy in capitals on the lines below)

Name..
Address..
.... Tele..........................

Please cut out and send this form to: Classified Department, Dragon User, 12 —13 Little Newport St, London WC2H 7PP.

CLASSIFIED ADS

Design or chance

***Gordon Lee** issues a creative challenge to budding wallpaper artists*

THIS month by way of a change the competition is departing from its usual format in favour of something a bit more creative and artistic. In the forthcoming months we will also be including something within the scope of any beginners who have previously fought shy of entering. So, come on all you Dragon users, now's your chance to try for some of those prizes — and don't worry, all you computer masochists, there will still be lots of toughies for you!

Very simply, the competition this month is to design an interesting visual display using one of the high resolution screens of the Dragon. To give you an idea of the sort of thing we're after, try the listing given on this page. As you can see, this program is quite compact, and yet, when run, it is surprising that such an intricate and yet logical design could be produced from this simple process.

This economy of programming will be one of the points that we will be looking for in the entries, which should utilise one of the high resolution screens (PMODE 0 to 4).

What we do not require is a lengthy program which, for example, has virtually every pixel or line drawn and listed within the program itself. You should aim for maximum effect from minimum program!

Your entry must be 'predictable', in the sense that when run it will produce an identical result each time, ie not a series of randomly placed circles or rectangles. It is quite in order to use the random feature to control minor details such as colour variations, but the overall design should be 'fixed'.

To enter the competition, send your finished program on a cassette only (no discs please), together with a listing (if possible) and any other documentation relating to your entry. There should be only one program on each cassette, placed at the beginning of the tape — though you may include a duplicated back-up copy on the other side. Ensure that your name is marked on the cassette label and, if you would like your tape returned, please enclose a stamped, self addressed envelope.

Finally, here is an analysis of the sample program given with this competition. It is based (very loosely) on Pascal's Triangle, a mathematical curiosity which will be dealt with on a future competition page.

As regards the pattern, imagine a rectangular grid 256 squares across by 192 squares down. In each of the squares along the top and left hand edges write in the number 1. Now, starting at the left hand end of the second row, write in each empty square to the immediate left. Continue along the row in this way to the right hand end and then repeat the procedure with each row in turn down to the bottom.

Of course, if we were to actually do this the numbers on most of the squares would soon become very large; however, if we merely coloured green each square with an odd number in it, and coloured black each square with an even number, the result would be the same as that shown on the screen. The simplicity of the program is dependent on the use of the PPOINT command to test the colour of the pixels above and to the left of each location, and using this information to decide whether to draw a green or black pixel. The actual numbers are not themselves evaluated, merely whether they are odd (black) or even (green).

That's it in a nutshell, so now it's up to all of you budding William Morrises to come up with something of your own!

```
10 PMODE4:PCLS:SCREEN1,0
20 LINE(0,0)-(255,0),PSET
30 LINE(0,0)-(0,191),PSET
40 FOR Y=1 TO 191
50 FOR X=1 TO 255
60 P1=PPOINT(X-1,Y):
   P2=PPOINT(X,Y-1)
70 IF (P1+P2)/2=INT
   ((P1+P2)/2)
   THEN PSET(X,Y,0)
   ELSE PSET(X,Y,1)
80 NEXT:NEXT
90 GOTO 90
```

Prize

"WOTCHA" said Microvision's main man. "We're marketing a few new programs in the new year." "Oh yeah" we yawned, acting blase. "How many?" "Seventeen" he said. So straight away we fell off our chair and touched him for a box of prize yummies. Seventeen is The Random Number, so our random choice was CATACOMB CRISIS. We have twenty Crises to give away to our January prizewinners.

Rules

Right. We're having a tiny departure from tradition. Because we realise that not everyone can easily supply a screen dump of their entry, we're allowing contestants to send in cassettes (not discs) as an alternative. Please put in an SAE if you want Gordon to send your cassette back.

The rest is as usual. Please remember to enclose a listing of your program wherever possible, your name and address, mark your envelope JANUARY COMPETITION, don't forget to post it and ..... YES! You thought we'd forgotten, didn't you? A TIEBREAKER. Complete the phrase 'If seventeen is the Random Number, then ...'.

October winners

The winners of Blaby Computer Games's BOULDER CRASH in the October competition are: G. R. Barber of Sutton Coldfield, Christopher James of Tarxien, S. A. Siddiqui of Chiswick, O. Friis Nielsen (not his mum) of Denmark, E. C. Hastead of Erith, F. J. Taylor of Middlesborough, S. Friedland of York, M. Owens of Pontypridd, Mike Johnson of Newcastle, Keith David of Crawley, Denis O'Mulloy of Comberton, A. Thomas of Staplehurst, J. Hewitt of Hixon, D. Denman of Twickenham, Terry Potter of Chiseldon, Michael Graham of Bangor, Mark Draven of Lincoln, Rolf Michelsen of Norway, Paul Weedon of Wooton Under Edge, and Mark Heaps of Arborfield.

One or two pretty good tiebreakers, but the favourite comes from S. A. Siddiqui:

"Look our for falling rocks, but don't worry ... BP will find you after a few million years."

Solution

OK, OK you asked for it, and if it makes you all feel crawlingly inferior, guilt-ridden, muddles, diddled and puzzled, serves ya all right.

See page 5.